Dix-septième colloque GRETSI, Vannes, 13-17 septembre 1999

Opérateur matériel d'étiquetage de régions temps réel et flot de données

Philippe LAMATY [1], Didier DEMIGNY [2]

[1]Service E/EEE/LO
Aérospatiale Missiles ; 8, rue Le Brix ; 18020 BOURGES Cedex ; France

[2]Equipe Traitement d'Images et du Signal
ENSEA/UCP ; 6, av. du Ponceau ; 95014 CERGY-PONTOISE Cedex ; France

philippe.lamaty@missiles.aerospatiale.fr, demigny@ensea.fr

Résumé – Nous proposons un algorithme d'étiquetage de régions en vue d'une intégration matérielle temps réel et flot de données. Nous exposons l'algorithme étudié et les différentes méthodes de réduction des ressources mémoires. Une de ces méthodes met en oeuvre une compression de type RLE (Real Length Encoding) avant de réaliser l'étiquetage de régions. L'étiquetage est ensuite effectué sur les données compressées.

Abstract – We propose an regions labeling algorithm in order to design a real-time and data-flow hardware architecture. We show the studied algorithm and the differents solutions for reduce the memory resources. A solution make a Real Length Encoding compression before realized the regions labeling. The labeling is doing on the compress datas.

1. Introduction

L'étiquetage de régions, ou étiquetage en composantes connexes, est une opération fondamentale du traitement d'images (Fig. 1). Ce traitement affecte un numéro d'identification, ou étiquette, à chaque composante connexe d'une image binaire [2, 7]. Cet opérateur a fait l'objet de nombreuses études sur les aspects algorithmiques et architecturaux qui ont mis en évidence deux types d'exécution des traitements : parallèle ou séquentielle. Dans le cas parallèle, l'itération de traitement utilise un parallélisme massif non compatible avec des systèmes embarqués [3, 6, 8]. Pour le cas séquentiel, les images sont traitées ligne par ligne [1, 5] et nécessitent deux passes. Ce double balayage a l'inconvenient de contraindre le traitement à mémoriser une image intermédiaire.

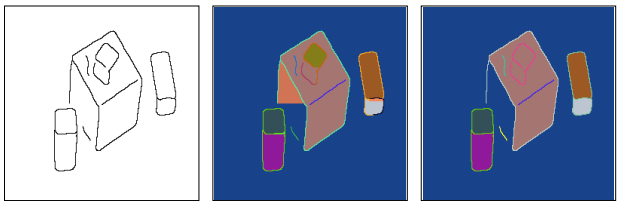


Fig. 1 : image contour (a) ; image des pré-étiquettes (b) ; image des étiquettes (c)

Dans la suite de cet article, nous allons décrire d'une part, l'algorithme d'étiquetage de régions qui nous a servi de base lors de notre étude et, d'autre part, les différentes méthodes d'optimisation des ressources mémoires. Les algorithmes présentés sont relatifs à un traitement 4 connexes. Ils pourront facilement être étendu à 8 connexités.

Par convention, l'image E définie l'entrée d'un traitement, I une image intermédiaire, et S l'image de sortie. La notation S[i, j] décrit le pixel à la position (i, j) dans l'image S.

2. Etiquetage séquentiel de régions

L'étiquetage des régions n'est pas un traitement local, car le voisinage d'un point de l'image n'est pas suffisant pour lui affecter un numéro d'identification. L'algorithme étudié se base sur les travaux de A. Rosenfeld et nécessite deux balayages de l'image [1].

La première passe est définie comme l'opération de pré-étiquetage (Fig. 1b). Elle utilise un automate en L inversé pour étiqueter les pixels en fonction de leur voisinage local (Fig. 2, Algo. 1). En parallèle, une table de correspondances est créée pour résoudre les équivalences dans le cas ou une région est décrite par plusieurs étiquettes (Algo. 2).

Fig. 2 : automate en L inversé

La deuxième passe, ou étiquetage final, utilise la table des correspondances pour résoudre les équivalences (Fig. 1c). Ce traitement permet d'affecter une étiquette unique à chaque région (Algo. 4).

Une mise à jour des données de la table peut être effectuée entre les deux, ou pendant l'un des deux balayages (Algo. 3).

Pour s'affranchir des problèmes liés aux effets de bord de l'image, un miroir de un pixel est ajouté sur les bords haut et

gauche de l'image. Les pixels ajoutés sont l'inverse du bord de l'image (Fig. 3). Ceci impose à l'algorithme de recopier l'étiquette A ou B du masque ou à créer une nouvelle étiquette.

```
pour tous les pixels de l'image
   si E[Pr] /= E[A] et E[B] alors
      I[Pr] ← nouvelle_étiquette
      incrémente_de_1( nouvelle_étiquette )
   si E[Pr] = E[A] et E[B] alors
      si I[A] < I[B] alors
         I[Pr] ← T[I[A]]
      sinon
         I[Pr] ← T[I[B]]
      fin si
   si E[Pr] = E[A] alors
      I[Pr] ← T[I[A]]
   si E[Pr] = E[B] alors
      I[Pr] ← T[I[B]]
   fin si
fin pour
```

Algo. 1 : pré-étiquetage

```
pour tous les pixels de l'image
   si E[Pr] /= E[A] et E[B] alors
      T[I[Pr]] ← T[I[Pr]]
   si E[Pr] = E[A] et E[B] alors
      si I[A] < I[B] alors
         T[I[Pr]] ← T[I[A]]
      sinon
         T[I[Pr]] ← T[I[B]]
      fin si
   fin si
```

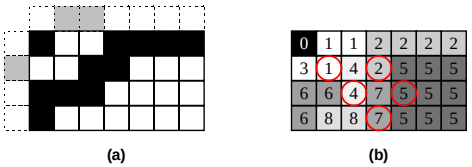
Algo. 2 : création de la table d'équivalences T

```
pour i de 0 à dernière_étiquette
   k ← i
   tant que T[k] ≠ k faire
      k ← T[k]
   fin tant que
   T[i] ← k
fin pour
```

Algo. 3 : mise à jour de la table d'équivalences

```
pour tous les pixels de l'image
   S[i, j] ← T[I[i, j]]
fin pour
```

Algo. 4 : étiquetage final



(a)

0	1	1	2	2	2	2
3	1	4	2	5	5	5
6	6	4	7	5	5	5
6	8	8	7	5	5	5

(b)

Création d'une équivalence

Table de correspondances avant et après mise en forme

label	0	1	2	3	4	5	6	7	8
avant	0	1	2	1	2	5	4	5	7
après	0	1	2	1	2	5	2	5	5

0	1	1	2	2	2	2
1	1	2	2	5	5	5
2	2	2	5	5	5	5
2	5	5	5	5	5	5

(c)

Fig 3 : image contour + bord virtuel (a) ; image des pré-étiquettes (b) ; image des étiquettes (c)

Une réalisation matérielle flot de données impose de doubler la mémoire de stockage et la table d'équivalences. Un « ping-pong » permet d'accéder à une mémoire en lecture pendant que les données sont écrites dans la seconde. Les accès lecture/écriture sont inversés à chaque nouvelle image.

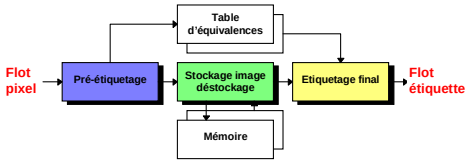


Fig 4 : architecture de l'opérateur d'étiquetage de régions

3. Optimisation des ressources

Nous allons maintenant détailler les méthodes de réduction de la mémoire de stockage et de la table des équivalences.

3.1 Réduction de la mémoire de stockage

Une réduction de cette mémoire de stockage peut facilement être obtenue par compression de l'image des pré-étiquettes (Fig. 5). Nous avons utilisé une compression Real Length Encoding (RLE) qui donne des résultats corrects et reste simple à réaliser matériellement (Fig. 6).

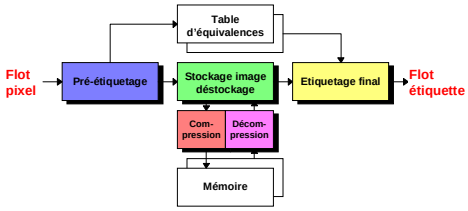
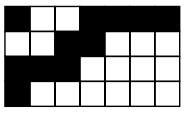


Fig 5 : architecture de l'opérateur d'étiquetage de régions avec compression de l'image des pré-étiquettes



(a)

(1, 1) (2, 0) (4, 1)
(2, 0) (2, 1) (3, 1)
(3, 1) (4, 0)
(1, 1) (6, 0)

(b)

(N, P)
Valeur du pixel
Nombre de répétition

Fig 6 : exemple de compression RLE

Une autre solution, beaucoup plus avantageuse, est de stocker l'image contour à la place de l'image des pré-étiquettes (Fig. 7). La deuxième étape de l'algorithme se retrouve totalement modifiée. Elle est équivalente au traitement de l'étape 1 (Algo. 1) sans création de table d´équivalences. Par contre, cette dernière est utilisée dans le cas de la création d'un nouveau label pour connaître l'identifieur de la région (Algo. 5).

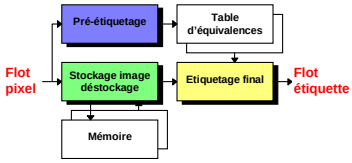


Fig. 7 : architecture de l'opérateur d'étiquetage de régions avec stockage de l'image contour

Une optimisation supplémentaire peut, là encore, être obtenue par compression de l'image contour.

```
pour tous les pixels de l'image
   si E[Pr] ≠ E[A] et E[B] alors
      S[Pr] ← T[nouvelle_étiquette]
      incrémente_de_1( nouvelle_étiquette )
   si E[Pr] = E[A] et E[B] alors
      S[Pr] ← S[A]   // ou S[Pr] ← S[B]
   si E[Pr] = E[A] alors
      S[Pr] ← E[A]
   si E[Pr] = E[B] alors
      S[Pr] ← S[B]
   fin si
fin pour
```

Algo. 5 : étiquetage final « modifié »

3.2 Réduction de la table d'équivalences

La réduction de la table d'équivalences est obtenue par prédiction des étiquettes. Cette méthode utilise un automate en L prédictif [4, 5] (Fig. 5). Le principe de prédiction est d'attendre le traitement de N pixels lors de la détection d'une nouvelle étiquette pour savoir s'il s'agit réellement d'une nouvelle étiquette. Dans le cas d'une réalisation matérielle le nombre de prédiction N sera limité.

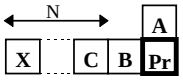


Fig. 8 : automate en L prédictif

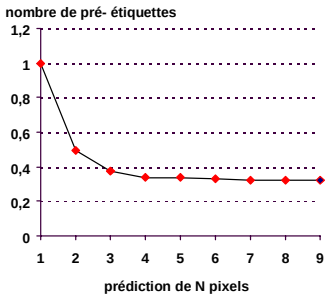


Fig. 9 : variation du nombre d'étiquettes en fonction de la prédiction

Remarque : La courbe de variations du nombre d'étiquettes (Fig. 9) est la moyenne des résultats obtenus sur plusieurs images du GDR 134. Pour chaque image, le nombre d'étiquettes est normalisé par rapport au nombre maximum de pré-étiquettes.

3.3 Réduction globale

Cette méthode utilise une compression de l'image contour et réalise les traitements sur les données compressées (Fig. 10). L'avantage de cette méthode est l'augmentation de la taille de prédiction des étiquettes, et la compression de l'image. L'inconvénient est la complexité des traitements des données compressées. La compression utilisée est de type « real length encoding (RLE) », elle permet un codage simple et efficace des images contours. Le problème de cette compression est une forte dégradation du taux de compression pour les motifs de type damier (succession de pixels fond/contour).

La compression RLE utilisée pour cette réduction compte le nombre de répétitions du motif M = {E[i, j-1], E[i, j]} sur deux lignes. Une donnée de compression est donc décrite par le nombre de répétitions N du motif M. C'est ce dernier qui est utilisé par l'algorithme de pré-étiquetage et d'étiquetage final. Une ligne dont la valeur des pixels est l'inverse de la ligne inférieure est ajoutée sur le bord supérieur pour simplifier l'algorithme de pré-étiquetage et supprimer les problèmes d'effets de bord.

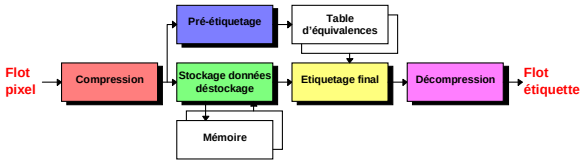


Fig. 10 : architecture de l'algorithme d'étiquetage de régions avec compression de l'image

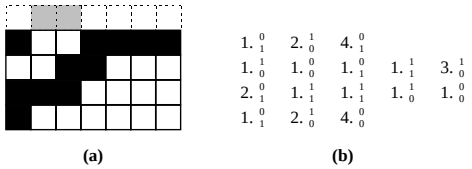


Fig 11 : exemple de compression RLE sur deux lignes ; image contour (a), données compressées (b)

Une information sur la détection d'une fin de ligne est ajoutée à la structure d'une donnée compressée. Cette information peut servir à déterminer trois états : début de ligne, dans la ligne, fin de ligne.

L'utilisation des données compressées pour réaliser les étapes de pré-étiquetage et d'étiquetage final permet de remplacer les lignes à retard nécessaires à la mémorisation d'une ligne de l'image par des files d'attente (FIFO). Leur profondeur est fonction du taux de compression obtenu sur les lignes de l'image. La compression RLE « modifiée » permet donc une réduction de la mémoire (Fig 11) de stockage image, mais aussi des ressources de traitement interne.

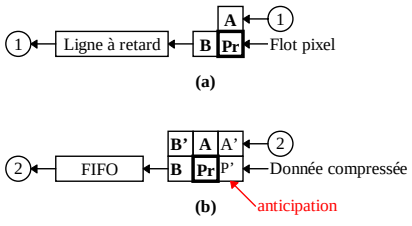


Fig 12 : pré-étiquetage ou étiquetage final ; ancienne architecture (a), nouvelle architecture (b)

Une donnée est écrite dans la file d'attente (FIFO), si la valeur du pixel B est différente du pixel Pr, ou si une fin de ligne est detectée. Une lecture est effectuée, si un début de ligne est détecté, ou si la valeur du pixel A est différente du

pixel A'. La file d'attente mémorise les étiquettes de B, qui servent au traitement de la ligne suivante.

Le motif d'« anticipation » $M_A = \{A', A\}$ permet aux algorithmes de réaliser une prédiction du label. Ce motif permettra de résoudre le cas ou les pixels Pr , P' et A' sont identiques et le pixel A est différent de A'. Dans ce cas, l'étiquette de A' est utilisée pour initialiser l'étiquette de Pr. Mis à part ce cas particulier de prédiction, l'algorithme de pré-étiquetage et d'étiquetage finale reste pratiquement identique à l'algorithme présenté au début de cette article.

Les traitements exécutés sur une image compressée font apparaître des « trous » de calcul qui peuvent servir à mettre à jour la table d'équivalences.

Un inconvénient de la compression d'image est qu'il faut recréer un flot vidéo comprenant une synchronisation horizontale et verticale, et un flot d'étiquettes.

Un autre désagrément est l'augmentation du taux de compression à partir d'un certain nombre de répétitions (Fig. 14). Cette dégradation est due à la compression RLE et aux images traitées, car le nombre de bits qui code le nombre de répétitions N intervient dans le taux de compression. La compression RLE optimale est ainsi obtenue pour un codage de N permettant de coder le nombre maximal de répétitions.

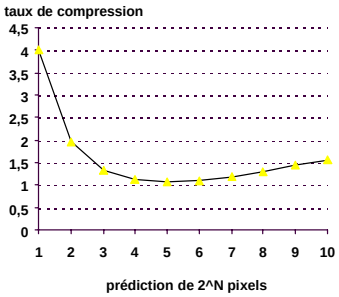


Fig 14 : variation du taux de compression en fonction du nombre de répétitions N

En résumé, nous montrons sur un exemple les gains mémoires obtenus sur l'ensemble des ressources nécessaires à la réalisation de cet opérateur. La mémoire de stockage n'est pas prise en compte, car pour des tailles images supérieures à 64×64 pixels elle devient difficilement implantable dans une puce.

Sans compression :

- 1 ligne à retard pour le pré-étiquetage, Largeur × Label,
- 2 tables d'équivalences, $2 \times 2^{Label} \times$ Label bits,
- 1 ligne à retard pour l'étiquetage final, Largeur × Label.

Avec compression :

- 1 ligne à retard pour la compression, Largeur × 1,
- 1 FiFo pour le pré-étiquetage de Profondeur × Label,
- 2 tables d'équivalences, $2 \times 2^{Label} \times$ Label,
- 1 FiFo pour le étiquetage final, Profondeur × Label,

Note : Label définit le nombre de bits nécessaire à coder une étiquette.

Nous obtenons pour une image de Largeur = 1024 points, Label = 10 bits (1024 étiquettes), Profondeur = 256 :

Sans compression : 40960 bits.

Avec compression : 26624 bits.

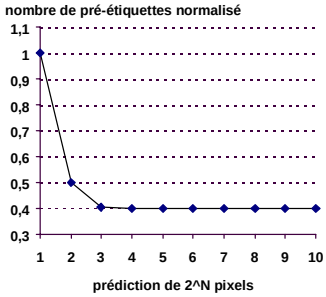


Fig 15 : variation du nombre d'étiquettes en fonction du nombre de répétitions N

4. Conclusion

Nous n'avons pas traité dans cet article l'influence des pré-traitements sur les ressources. Mais des travaux montrent qu'une réduction du nombre d'étiquettes peut être obtenue par un opérateur d'« amélioration » des images contours [5].

Nous avons réalisé une architecture comprenant la compression et le stockage de l'image contour (Fig. 10). Pour dimensionner les ressources mémoires, nous avons simulé l'opérateur d'étiquetage sur différentes images. Les résultats de ces simulations nous ont permis de constater l'influence de la prédiction sur la taille de la table d'équivalences. Cet algorithme a été intégré dans un ASIC en technologie « embedded gate array » 0,35µm. Il fonctionne à 20 Mhz pour des images maximales de 1024x1024.

Références

[1] A.ROSENFELD, J. L. PFALTZ - « *Sequential operations in digital picture processing* », Journal of ACM, vol. 13, n° 4, pp.471-494, 1966

[2] A.ROSENFELD - « *Digital Picture Processing* », Academic Press, 1982

[3] J.M.CHASSERY - « *Deux algorithmes orientés parallélisme : courbes de niveau et étiquetage* », 5ème congrès RFIA, Grenoble, pp.541-548, Novembre 1985

[4] J.QUESNE, D.DEMIGNY, J.DEVARS, J.P.COCQUEREZ - « *Architectures temps réel pour la fermeture des contours et l'étiquetage des régions* », 8ème congrès de reconnaissance des formes et intelligence artificielle, Lyon, pp.65-80, Novembre 1989

[5] J.QUESNE- « *Thèse: Vision robotique: Architectures data-flow pour le traitement des images en temps réel* », pp. 65-82, Janvier 1992

[6] E.MOZEF, S.WEBER, J.JABER, E.TISSERAND - « *Architecture dédiée à l'algorithme parallèle $O(n.log_2 n)$ d'étiquetage de composantes connnexes* », GRETSI Symposium on Signal and Image Processing, Juan les Pins, pp.83-89, Septembre 1995

[7] J.P.COCQUEREZ, S.PHILIPP - « *Analyse d'images : filtrage et segmentation* », Masson, pp.61-63, 1996