B] Traitement par segment (Mathieu)

a. Introduction

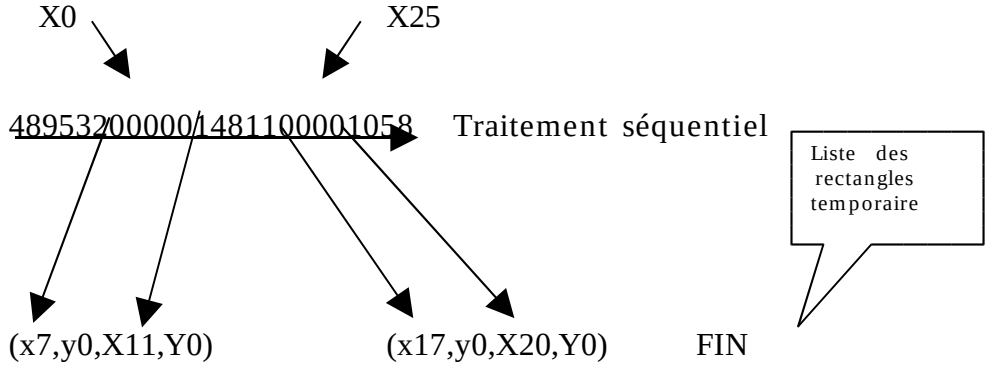
L'image à traiter est codée comme une matrice à deux dimensions afin de trouver les rectangles optimaux, il faudrait donc, dans un premier temps, détecter les segments (ou bandes) sur chaque ligne. Ainsi, dans un second temps, le traitement pourrait considérer les bandes détectées à la ligne précédente avec les bandes détectées à la ligne courante dans le but de la fusionner. Ce traitement se décompose donc en deux phases.

b. Analyse/Conception

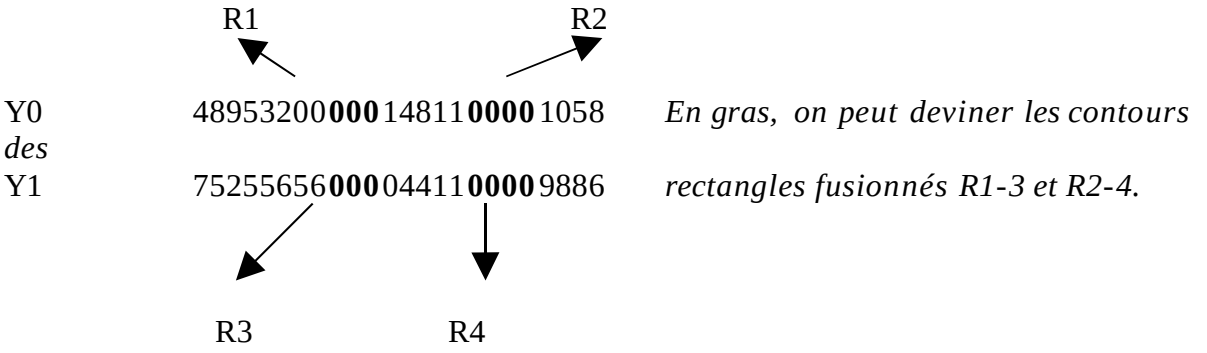
La matrice est analysée ligne par ligne : il s'agit d'un traitement séquentiel. Une image comporte Y lignes, une ligne comporte X octets. La valeur de chaque octet représente la couleur en niveau de gris (256 valeurs 0 à 255).

Le traitement comporte plusieurs étapes, d'abord il doit détecter une bande.

Lors de son parcours de la ligne, le traitement recherche la couleur à isoler (par ex : 0 couleur noire), au premier octet correspondant a cette valeur, il enregistre sa position dans la matrice (x,y) et déclare qu'une bande vient d'être ouverte après quoi il poursuit son parcours. Dés qu'il rencontre une valeur autre que la valeur à isoler, il déclare la bande comme fermée , considère sa position dans la matrice (X,Y) et enregistre la bande (x,y,X,Y) dans une liste de rectangles stockée en mémoire (Rq :une bande est un rectangles particulier).



La détection n'est qu'une étape, car il fait dégager des rectangles ayant une aire optimale sur l'image. Parallèlement, un traitement de Fusion vient analyser la possibilité de fusionner deux bandes entre elles. En fait, ce traitement intervient lorsqu'une bande est trouvée. Les bandes comparées seront assemblées si elles répondent a des critères prédéfinis de positionnement l'une par rapport à l'autre (cf. Les cas de l'Analyse - Fusion).



Analyse « simplifiée »pas à pas du traitement sur l'exemple :

En R1 : Une bande a été détectée (R1). La liste est vide. Pas de fusion possible. Ajouter R1 dans la liste. Continuer...

En R2 : Une bande est détectée (R2). La liste contient R1. Fusion lancée. R1 et R2 incompatibles selon les cas définis. Ajouter R2 dans la liste. Continuer...

En R3 : Une bande est détectée (R3). La liste contient R1 et R2. Fusion lancée. Parcours séquentiel de la liste. R1 et R3 compatible, création du rectangle fusionné R1-3, destruction de R1 et R3, ajout dans la liste de R1-3 et des « rectangles miettes » résultant de la fusion (ici R1,R3 sont « cassés » en deux) on rajoute donc R1 et R3. R2 et R1-3 incompatibles. Continuer...

En R4 : Une bandes est détectée (R4). La liste contient R2 et R1-3. Parcours séquentiel de la liste. R2 et R4 compatibles, création du rectangle fusionné R2-4, destructions de R2 et R4, ajout dans la liste de R2-4, pas de « rectangles miettes ». Fin.

c. Cas de fusion

Il existe 8 cas lors d'une analyse fusion entre 2 rectangles ou bandes (R1 et R2). Chacun de ces cas nécessite que R1 s'ouvre avant R2 et que R1 se ferme après R2 soit (r1.Y>=r2.Y-1) et (r1.y<=r2.y-1) en C/C++.

Dans le schéma ci-dessous, 0 représente la couleur à isoler et 1 tout autre couleur, en gras le rectangle fusionné, les bandes soulignées sont les rectangles miettes:

Cas de base :

		x X	
R1	:	1**00000000**1	(r1.x==r2.x et r1.X==r2.X)
R2	:	1**00000000**1	

Cas a:

		x X	
R1	:	1**000000**<u>00</u>1	(r1.x==r2.x et r1.X>r2.X)
R2	:	1**000000**111	

Cas b:

		x X	
R1	:	1<u>00</u>**000000**1	(r1.x<r2.x et r1.X==r2.X)
R2	:	111**000000**1	

Cas c:

		x X	
R1	:	1**0000**11111	(r1.x<r2.x et r1.X==r2.X)
R2	:	1**<u>0000</u>**<u>0000</u>1	

Cas d:

		x	X	
R1	:	1111**00000**1		(r1.x>r2.x et r1.X==r2.X)
R2	:	1000**00000**1		

Cas e:

		x	X	
R1	:	1111**00**1111		(r1.x>r2.x et r1.X<r2.X)
R2	:	1000**00**0001		

Cas f:

		x	X	
R1	:	0**0000000**01		(r1.x<r2.x et r1.X>r2.X)
R2	:	1**0000000**11		

Cas g:

		x	X	
R1	:	11**000**0001		(r1.x>r2.x et r1.X>r2.X et r2.X>=r1.x)
R2	:	00**000**11111		

Cas h:

		x	X	
R1	:	0000**00**1111		(r1.x<r2.x et r1.X<r2.X et r2.x<=r1.X)
R2	:	1111**00**0001		

d. Optimiser la vitesse d'exécution

L'intérêt de ce programme est, bien sûr, de fusionner un maximum de rectangles et de segments (bandes), néanmoins ces objets « empilés » dans la liste finissent par ralentir le temps d'exécution de façon alarmante, en effet pour chaque tentative de fusion, un parcours séquentiel de la liste est enclenché, or nous avons vu que deux segments ou rectangles qui fusionnent génèrent dans le pire des cas trois rectangles.

Il est donc nécessaire d'opérer un transfert des rectangles définitifs, les rectangles optimaux. Dans un premier temps, les rectangles susceptibles d'être encore fusionnables restent dans la liste temporaire (ou tampon) . Dans un second temps, lors d'un parcours de cette liste pour une fusion, un test vérifie que le rectangle est optimal et si tel est le cas, il le déplace vers la liste finale.

Schéma du mécanisme avec les même hypothèses de traitement que précédemment :

Y0	4895320**000**14811**0000**1058	*En gras, on peut deviner les contours*
Y1	7525565**000**04411**0000**9886	*des rectangles fusionné.*
Y2	969558099999900984135700	
Y3	814612179800238951111245	

Lors du parcours de Y3, on sait que les rectangles fusionnés sont définitifs. Concrétement, leur coordonnée de ligne Y (coin droit en bas) est inféreur strictement à la ligne parcourue couramment par le balayage séquentiel prinicpal.

e. Signalétique de déboggage

Cette partie explique succinctement les signalétique que j'ai adopté lors des affichages de déboggage (par défaut désactivé, drapeau = 0) :

$2,3,4,5$ *coordonnée d'une bande ou segment.*

$2,3,4,9@ *coordonnée d'une bande forcée à la fermeture (fin de ligne).*

Cour(1,1,5,1) r(1,2,5,2) =>(1,1,5,2) *fusion réussie avec le rectangle comparé Cour et le rectangle comparant r.*

ERR_NORM : (2,3,1,1) *erreur coordonnée incohérente.*

[2,3,5,11] T: 18 F: 1425 *Transfert d'un rectangle optimal dans la liste finale, il y a couramment 18 bandes ou rectangles dans la liste tampon et 1425 dans la liste finale.*

f. Analyse mes algorithmes principaux mis en œuvre dans le traitement

Les algorithmes seront expliqués selon le séquencement de l'exécution du programme. Ainsi, d'abord on charge en mémoire l'image du bitmap lu à partir du fichier .RAW. Ensuite, on traite l'image séquentiellement dés qu'une bande est trouvée, on lance le traite d'optimisation, qui lance le traitement de fusion. Enfin on détruit l'image mémoire.

1) Algorithme de chargement de l'image : ChargerBitmap

paramètres nécessaires : Nom du fichier , Taille image en X et en Y
Valeur de retour : un pointeur sur le tableau de pointeurs des lignes

Ouverture du fichier .RAW en mode binaire (sinon mode est ascii par défaut)
Création mémoire de l'image, il s'agit d'un tableau de pointeur qui représente les lignes pointant elles-mêmes sur un tableau de caractères (chaque caractère représente la valeur décimale de la couleur en mode 256 couleurs).On initialise au nombre de ligne Y.

Pour toute ligne Y de la matrice
    Création mémoire d'un tableau de pointeur sur caractère de taille X.
    Lecture d'une ligne fichier et affectation au tableau (lecture tamponnée par ligne)
Fin de pour tout

Renvoi d'un pointeur pointant sur le tableau de pointeur sur lignes

2) Algorithme de détection des bandes : DetecteurdeBande

paramètres nécessaires : Nom du fichier , Taille image en X et en Y , valeur décimale de la couleur à isoler en rectangles optimaux, la liste finale où stoker les rectangles optimaux
Valeur de retour : aucune (procédure)

On déclare une liste tampon (ou temporaire)
On déclare un fichier (un tableau de caractères, il s'agit du nom du fichier)
On déclare un booléen Bande, qui fera office de drapeau indiquant si une bande est ouverte ou non

Parcourir les lignes de la matrice
Parcourir les colonnes de la matrice

Le bit courant est de la couleur à isoler et le drapeau indique qu'aucune bande à était encore ouverte : ON A TROUVE LE DEBUT D'UNE BANDE

- Sauvegarde des coordonnées (x et y courant) indiquant le début de bande
- On indique qu'on a trouvé une bande (drapeau Bande activé)
- Ajouter au compteur de bandes détectées qu'on vient d'en trouver une nouvelle (incrémentation)

Le bit courant n'est de la couleur à isoler et le drapeau indique qu'une bande est actuellement « ouverte » : ON A TROUVE LA FIN DE LA BANDE OUVERTE

- On indique qu'on a fermé la bande, il n'y a plus de bande courante (drapeau Bande désactivé)
- On instancit un objet avec les coordonnées sauvegardées précédemment indiquant le début de la bande et les cordonnées de fin , soit les coordonnées courante

- On lance l'analyse-fusion (cf. AnalyseFusion) avec ce nouveau rectangle afin de voir si déjà on ne peux pas le fusion avec un précédent

Fin de parcours des colonnes

Si une bande est encore ouverte alors qu'on arrive en fin de ligne on force la fermeture. Et on traite ce cas de la même façon que précédemment : ON A TROUVE LA FIN D'UNE BANDE

Fin de parcours des lignes .

Affichage statistiques pour information utilisateur.
On détruit la liste des rectangles.
On lance la destruction de l'image mémoire (cf. LibéreBitmap)

3) Algorithme d'analyse et de fusion des bandes/rectangles en rectangles : AnalyseFusion

paramètres nécessaires : La liste des rectangles finale, la liste tampon, le rectangle à fusionner, la ligne courante du parcours séquentiel de DetecteurDeBande
Valeur de retour : aucune (procédure)

On replace le pointeur rectangle courant de la liste tampon (cf. Structure de données) sur le début de la liste.
On déclare un booléen indiquant la « fusionnabilité » du rectangle (il stockera une valeur de retour)
On déclare un rectangle principal qui stockera le rectangle résultant de la fusion
On déclare aussi 2 rectangles qui stockeront les morceaux (« rectangles miettes ») résultant de la fusion dans certains cas

Tant que la liste des rectangles n'est pas vide IL Y A UN COMPARANT (le rectangle pointé par le pointeur courant n'est pas égal à null)

Le rectangle courant n'est pas déjà optimal (ou « fermé ») ?

OUI

L'ajouter dan la liste finale

Le supprimer de la liste tampon (en fait un déplacement)

Incrementer le compteur de fusion (servant dans ce cas principalement indicateur de visite de cette méthode)

NON

- Lancer la méthode de fusion (cf. Optimize) avec le rectangle à fusionner (le comparé) et le rectangle courant (le comparant) dans la liste tampon
- Si la valeur renvoyée par cette méthode (stockée dans le rectangle principal) indique qu'il est fusionnable :

    - On indique a un compteur qu'une fusion à lieu (incrément)
    - On redéfinit le rectangle comparé avec le rectangles principal
    - Si des « rectangles miettes » existent alors on les ajoutent tout de suite dans la liste tampon. (on sait qu'ils existent s'ils n'ont pas une valeur « signal » de non fusionnabilité cf. Optimize)
    - On supprime enfin le rectangle comparant, il a était « absorbé » par le nouveau rectangle crée

- On indique au pointeur courant dans liste tampon de référencer « le rectangle suivant du courant »
- On ajoute le rectangle comparé qui a était redéfini ou non (si effectivement réussi à le fusionner)

Fin de parcours de la liste tampon

4) Algorithme de fusion et d'évaluation des cas : Optimize

paramètres nécessaires : le rectangle/bande à comparer, le rectangle/bande comparant
Valeur de retour : le rectangle résultant de la fusion, 2 rectangles miettes résultant de la fusion (si tel est le cas)

On déclare un rectangle signal de « non-fusionnabilité », ce rectangle qui sera renvoyé par cette fonction indiquera alors à la procédure d'analyse-fusion qu'aucune modification au rectangle d'origine
à était faite. Ce signal sera aussi utilisé pour signifier à cette procédure si les rectangles miettes ont étaient générer ou non.
On initialise le rectangle de retour comme par défaut non-fusionnable, et on fait de même pour les rectangles miettes indiquant ainsi leur inexistence par défaut.

Si les cordonnées en Y du rectangle comparant indique qu'il était « à l'intérieur » du rectangle comparé : ILS SONT PEUT-ETRE FUSIONNABLES SELON DES CAS DEFINIS

- Considérer chaque cas de fusion (cf. Cas de fusion). Si un cas est avéré alors initialiser le rectangle de retour principal comme le rectangle-fusion des rectangles d'origine . Initialiser selon le cas les rectangles miettes avec « les restes » des rectangles comparé et comparant.

- Si aucun la disposition de n'est pas répertoriée alors renvoyer le rectangle de retour inchangé depuis sa déclaration (soit le rectangle de non-fusionnabilité)

Fin de la fonction

5) Algorithme de destruction de l'image mémoire : LibéreBitmap

Pour toute les lignes Y de la matrice

Détruire le tableau de pointeur des lignes pointant sur les tableaux de caractère

Fin de parcours de lignes.

Détruire le pointeur pointant sur le tableau de pointeur des lignes désormais vide

e. Problèmes principaux et solutions trouvées

Le premier problème, auquel j'ai été confronté est les erreurs lors d'allocation de mémoire pour la matrice image. Ce problème a été résolut avec un changement de compilateur (de Borland c++ 3.0 vers Microsoft Visual c++ 5.00 Pro) et la modification des directives de compilation (mode mémoire = Huge) afin d'avoir la permission d'allouer sur toute la mémoire utilisateur.

Un oubli a également engendré un problème dans la détection des bandes, en effet les bandes encore "ouvertes" en fin de ligne ne peuvent être prolongées sur la ligne suivante, et doivent être forcée à la "fermeture".

Durant la phase de développement de l'algorithme de fusion, un rectangle qui devait être fusionné peut être disjoint, l'intersection des deux rectangles comparés (la partie fusionnée) et les "miettes" de rectangles résultantes. Je me suis rendu compte après relecture de la thèse et entretien avec M. Ramdamne que ces "miettes" ne devaient être réduites seulement a l'aire différence entre l'aire totale des deux bandes et l'aire du rectangle fusionné; mais les miettes devaient être initialisées aux bandes à fusionner afin qu'elle puissent être utile pour une autre fusion.

Durant la phase de débugage et de tests, je me suis rendu compte que certaines bandes qui aurait du être fusionnées car elles répondaient à un cas de fusion précis, ne fusionner pas. Après maintes reprises du contenu du code des cas infructueuses, mon problème m'a rappeler un problème d'effet de bord que j'avais rencontrer en TP d'algorithmique. En effet, le problème ne concernait pas le fond mais la forme : c'est-à-dire le séquencement des cas dans la méthode. En fait, un cas correspondait à la configuration de fusion, une solution de fusion était donc proposée, mais cette dernier correspondait désormais à un autre cas ! La solution consiste a la mise en commentaire des cas qui faisaient "effet de bord".