N-803

IMPLANTATION D'UN LOGICIEL DE MORPHOLOGIE MATHEMATIQUE SUR CALCULATEUR PARALLELE PROPAL 2

S. BEUCHER - F. MARTIN

FONTAINEBLEAU MARS 1983

IMPLANTATION D'UN LOGICIEL DE MORPHOLOGIE MATHEMATIQUE SUR CALCULATEUR PARALLELE PROPAL 2

- Rapport Technique n° 1 -

S. BEUCHER, C.G.M.M. - F. MARTIN, E.T.C.A.

PRESENTATION GENERALE

L'Etablissement Technique Central de l'Armement et le Centre de Géostatistique et Morphologie Mathématique ont convenu d'étudier conjointement l'implantation sur un calculateur parallèle PROPAL 2 d'un logiciel de traitement des images basé sur les principes de la morphologie mathématique. Cette implantation doit être suivie par des études d'application dans les domaines du contrôle non destructif d'une part, de la biologie d'autre part. Le présent rapport a pour but de décrire les premières réalisations effectuées en ce sens, et notamment l'implantation des programmes de traitement morphologiques des images binaires.

A/ La Morphologie Mathématique

Le but essentiel de la morphologie mathématique est de donner les moyens d'obtenir des renseignements quantitatifs cohérents et significatifs sur des structures spatiales.

Pour cela, elle utilise un ensemble d'opérations élémentaires appliquées à des images binaires ou de teinte. Les opérations transforment l'image analysée en modifiant la valeur de chaque pixel en fonction d'un voisinage encore appelé "élément structurant".

Ces opérations élémentaires peuvent être concaténées afin d'obtenir des transformations de plus en plus complexes (cf. [1]). L'étape ultime d'un traitement morphologique consistant en une mesure effectuée sur l'image, le logiciel mis en place doit donc être capable d'effectuer ces mesures. Des considérations théoriques montrent que les mesures effectivement compatibles avec les principes de base de la morphologie mathématique sont peu nombreuses. Nous y reviendrons plus en détail dans la suite de ce rapport.

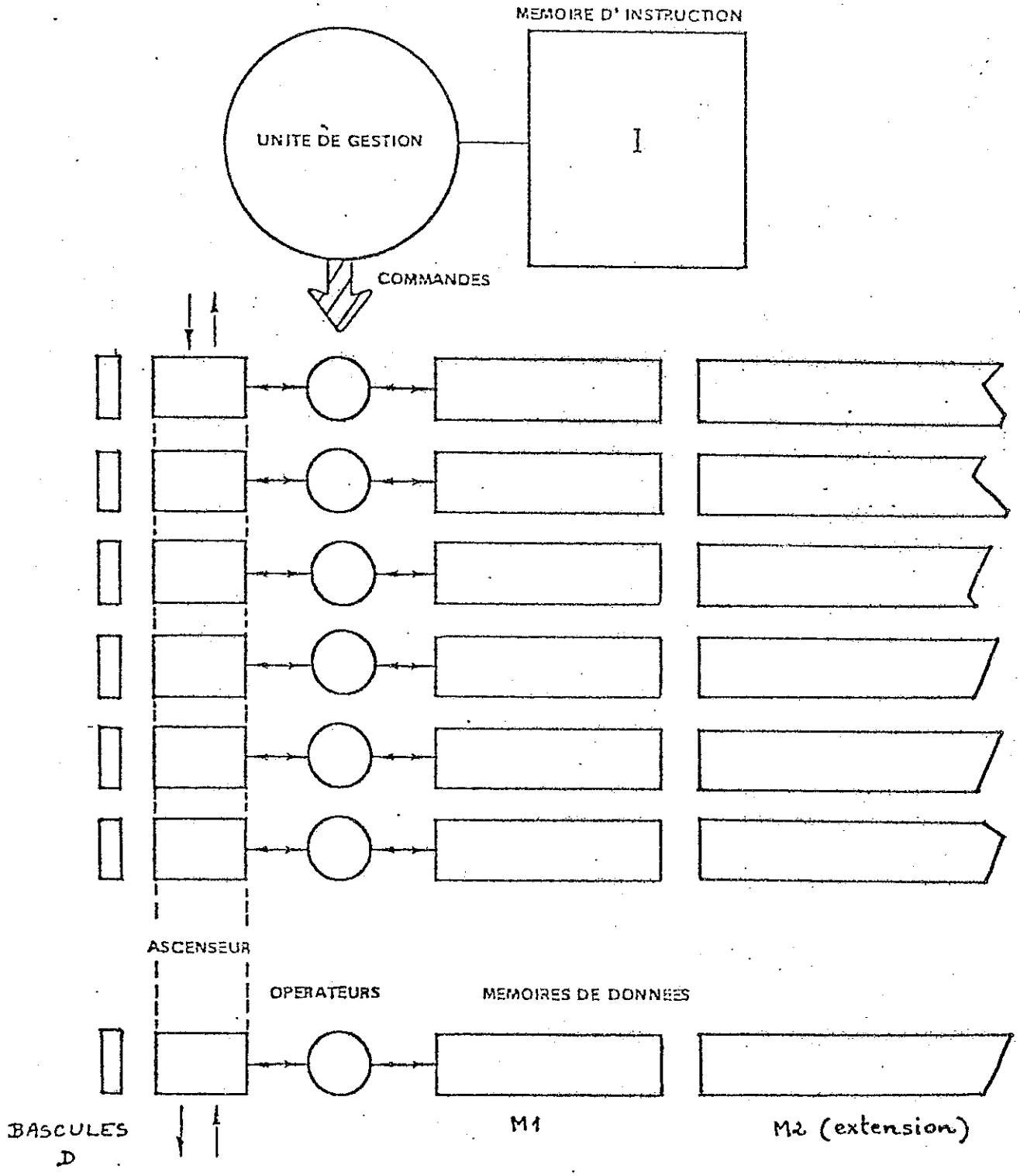


Figure 1 : Architecture de PROPAL 2

B/ Le calculateur PROPAL 2

PROPAL 2 est un ordinateur parallèle du type SIMD ("Single instruction Multiple Data Stream"). Cette machine a été conçue à l'ADERSA-GERBIOS et développée par la CIMSA. Cet ordinateur comprend 128 processeurs élémentaires (du moins dans la configuration que nous avons utilisée), une mémoire de travail de 256 bits associée à chaque processeur, et une mémoire d'extension (réservée au stockage) de 16 Kbits (par processeur). PROPAL 2 est une machine vectorielle ; chaque processeur travaille sur un élément du vecteur défini sur un nombre de bits variable (de 1 à 256). Les échanges de données entre les processeurs, ou avec l'environnement extérieur se font par l'intermédiaire d'un registre "Ascenseur". De plus, il est possible d'inhiber le fonctionnement de certains processeurs par l'intermédiaire d'un registre 1 bit dénommé bascule D. Ce registre joue également le rôle d'indicateur de débordement dans les opérations arithmétiques (Figure 1).

Cet ensemble est géré par une unité de gestion (U.G.). L'U.G. assure l'adressage des mémoires de travail et d'extension, commande les processeurs, gère les registres et, par l'intermédiaire d'une mémoire commune de 32 Kmots permet l'accès à un ordinateur-hôte (MITRA 125).

Le langage de programmation utilisé est un sur-ensemble de l'assembleur MITRA. On peut également utiliser un Fortran réduit pour les tâches implantées.

Le système d'exploitation de PROPAL 2 est supporté par le moniteur standard MITRA MMT 2 (cf. [2]).

C/ Réalisation des objectifs

Trois étapes successives ont été définies :

- la réalisation d'un logiciel de traitement des images binaires.

- la réalisation d'un logiciel de traitement des images de teintes.

- et enfin, l'étude des applications en C.N.D. et biologie.

En effet, si conceptuellement il n'existe aucune différence entre une transformation morphologique appliquée à une image binaire et la même transformation définie sur une image de teintes, on constate en pratique que l'utilisation des mêmes algorithmes pour ces deux types d'images pénalise fortement les transformations binaires du point de vue vitesse de traitement et occupation mémoire. D'autre part, il est généralement d'usage lors d'une étude morphologique de commencer par des traitements d'images de teintes qui permettront d'extraire les ensembles à analyser. Cette étape réalisée, il devient alors très rentable de posséder un logiciel uniquement dévolu aux images binaires. Cette façon de procéder réduit considérablement le temps de traitement.

Le présent rapport décrit dans le détail, les différentes parties du logiciel de traitement des images binaires. Les transformations mixtes (c'est-à-dire celles qui permettent d'obtenir une image binaire à partir d'une image de teintes, seuillage extraction de niveaux de gris, etc...) seront succintement introduites plus loin. Néanmoins, nous verrons que dans le logiciel binaire des possibilités de stockage d'images de teintes ont été prévues, ceci précisément afin d'assurer une mise en oeuvre simplifiée de ces transformations mixtes.

LE LOGICIEL DE TRAITEMENT DES IMAGES BINAIRES

A/ Présentation

Pour différentes raisons décrites ci-après, nous avons fixé la taille des images à traiter à 256 x 256 x 1 pour les images binaires et 256 x 256 x 8 pour les images de teintes. Cette taille réduite alors que la tendance en imagerie numérique est au traitement des images de 512 x 512 pixels, s'explique par la configuration PROPAL 2 que nous avons utilisée. De plus, ces images possèdent des caractéristiques très particulières. La première d'entre elles est que ces images sont échantillonnées suivant une grille non pas carrée mais hexagonale (le lecteur désirant savoir pourquoi ce maillage est plus intéressant que le maillage carré lorsqu'on utilise des transformations morphologiques pourra se référer à [5]). La dernière caractéristique

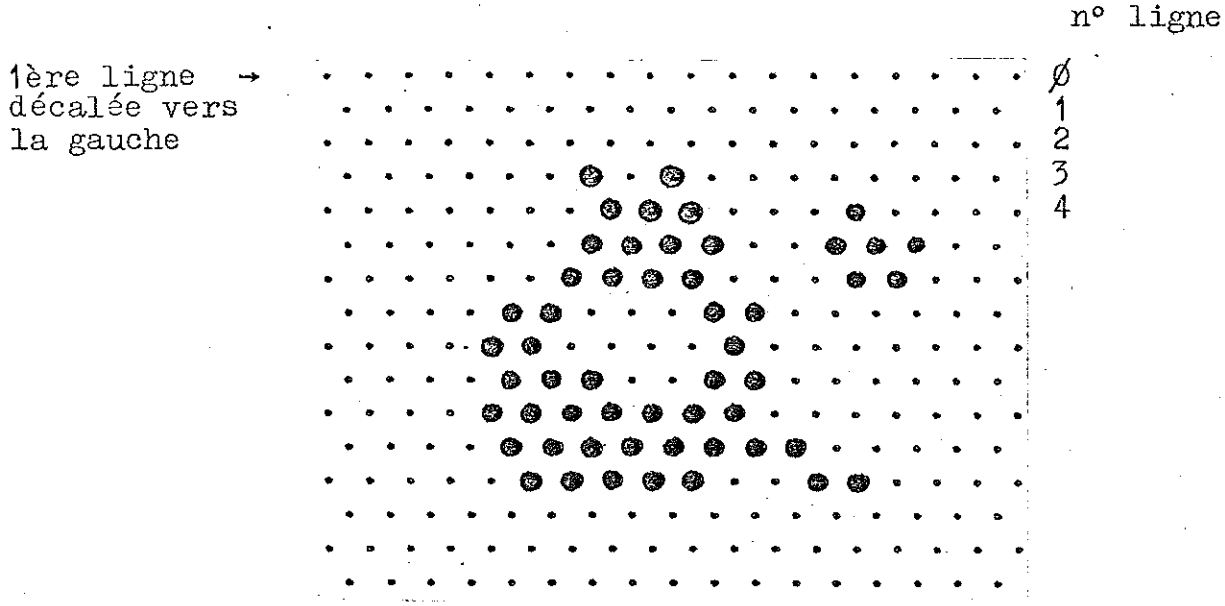


Figure 2 - Protocole d'échantillonnage des images.

est que la première ligne de l'image (c'est-à-dire la ligne ∅) est décalée vers la gauche (Figure 2).

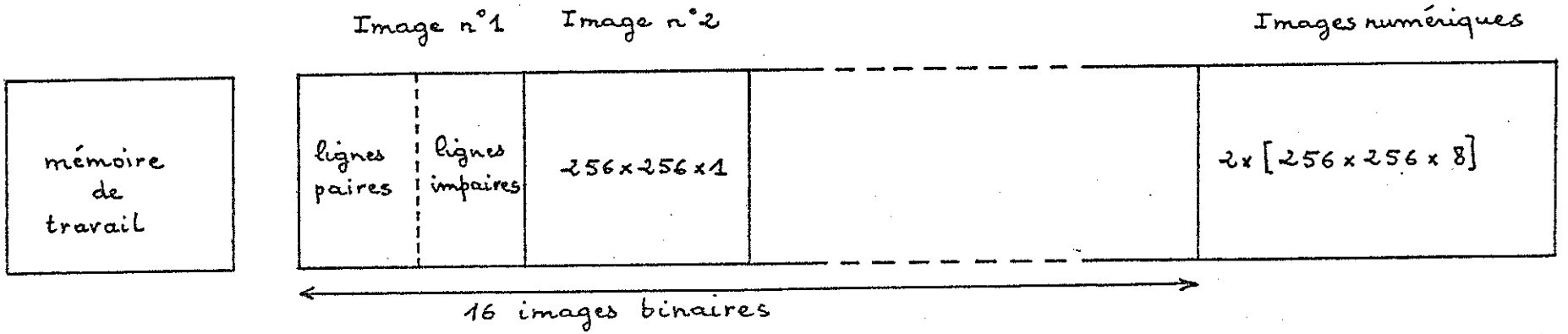
B/ Structure de la mémoire

On a vu que la mémoire de PROPAL 2 est composée de deux parties : la mémoire de travail et la mémoire d'extension. La mémoire de travail est utilisée uniquement pour effectuer les transformations morphologiques. La mémoire d'extension est une mémoire de stockage d'images. Un rapide calcul montre qu'il est possible de stocker dans cette mémoire 32 images binaires 256 × 256. En fait, la mémoire d'extension est séparée en deux parties : l'une réservée au stockage de 16 images binaires, l'autre au stockage de deux images de teintes 256 × 256 × 8. Cette disposition facilite la mise en oeuvre d'opérations mixtes ainsi qu'on l'a décrit plus haut.

En ce qui concerne la représentation interne d'une image, il a paru préférable de séparer les lignes paires et les lignes impaires afin d'éliminer tout problème de recouvrement entre zones de l'image et de limiter les échanges de données entre processeurs. Enfin, les processus d'entrée/sortie des images font que celles-ci apparaissent retournées dans la mémoire d'extension. La figure 3 résume ces différentes caractéristiques.

C/ Contraintes et conventions

Nous avons cherché le meilleur compromis possible entre souplesse d'utilisation et vitesse d'exécution. Ceci nous a conduit à n'utiliser la mémoire d'extension que pour le stockage des images. Ainsi l'utilisateur peut choisir de façon totalement libre les images d'entrée et de sortie d'une transformation. En particulier, il est possible d'écraser l'image initiale par l'image résultante. Eliminant ainsi tout stockage de résultat intermédiaire dans la mémoire d'extension, d'une part on augmente la vitesse de traitement (en effet les transferts Travail ⟷ extension sont très pénalisants) et d'autre part, on évite à l'utilisateur la surprise somme toute fort désagréable de voir une image utile détruite à son insu.



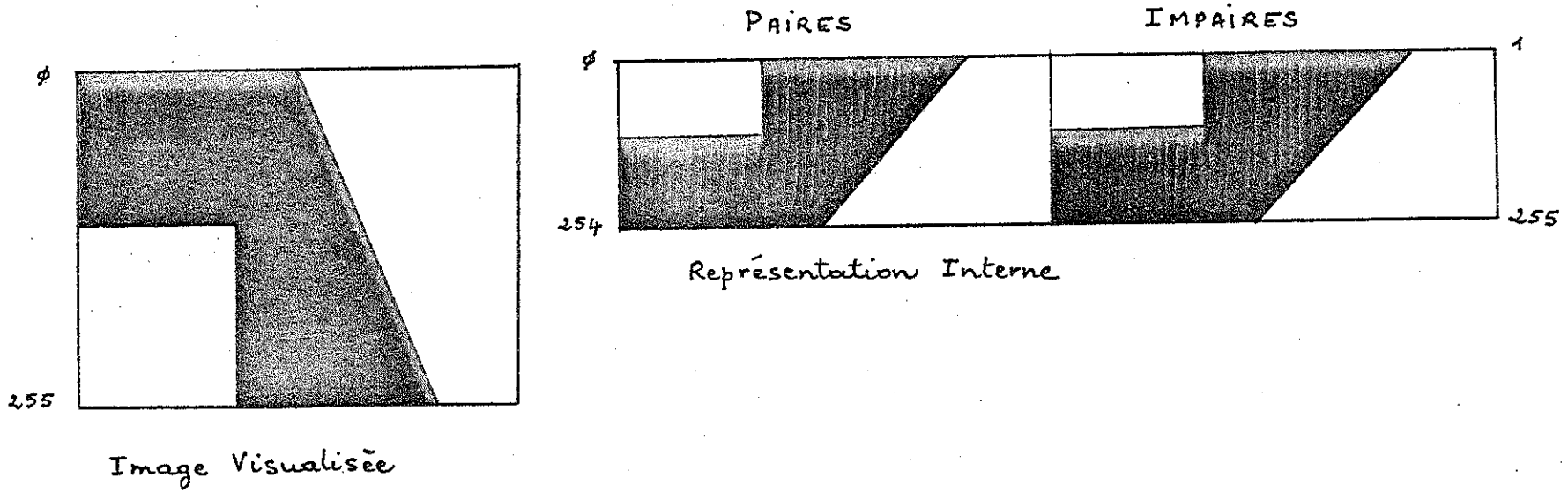


Figure 3 : Organisation de la mémoire d'extension

Une simplification importante nous a été dictée par l'expérience acquise depuis plusieurs années dans le traitement des images par la morphologie mathématique. Cette simplification consiste à n'utiliser pour les transformations de base que des éléments structurants définis sur un voisinage hexagonal de taille 1. Cette restriction n'est pas du tout gênante en pratique car la plupart des transformations élémentaires sur des voisinages de taille supérieure à 1 peuvent être décomposées en une suite de transformations de taille 1. Cette simplification apportée, il suffit donc de pouvoir mettre en relation chaque point de l'image avec ses plus proches voisins pour pouvoir effectuer l'ensemble des transformations qui nous intéressent. Nous verrons plus loin comment le fait de séparer les lignes paires et impaires nous permet de simplifier considérablement le problème. Mais avant cela, il convient de s'arrêter sur un point fondamental des transformations morphologiques : il s'agit des problèmes liés aux effets de bord. Décrivons succintement le problème : Supposons que nous effectuions une transformation à l'aide d'un élément structurant de taille 1 (peu importe la transformation à ce stade du raisonnement). Considérons un point situé sur le bord de l'image (Figure 4a). Il n'est pas possible de savoir quelles valeurs sont affectées aux points de son voisinage qui tombent à l'extérieur du champ.
Si, malgré cela, on désire pouvoir effectuer la transformation sur ce point, on doit obligatoirement définir arbitrairement une valeur à affecter à chacun des points extérieurs au champ de son voisinage. Par convention, nous avons choisi d'affecter la valeur $\emptyset$ aux points du voisinage qui tombent à l'extérieur du champ de l'image (Figure 4b). Tout se passe donc, lors d'une transformation, comme si on traitait une image 258 × 258 dont le bord serait systématiquement à $\emptyset$. Cette convention a des conséquences importantes qu'il convient d'avoir présentes à l'esprit. Ainsi, la dualité des transformations morphologiques n'est plus totalement respectée (ceci veut dire par exemple que l'érosion d'un ensemble et la dilatation de son complémentaire ne fournira pas le même résultat).

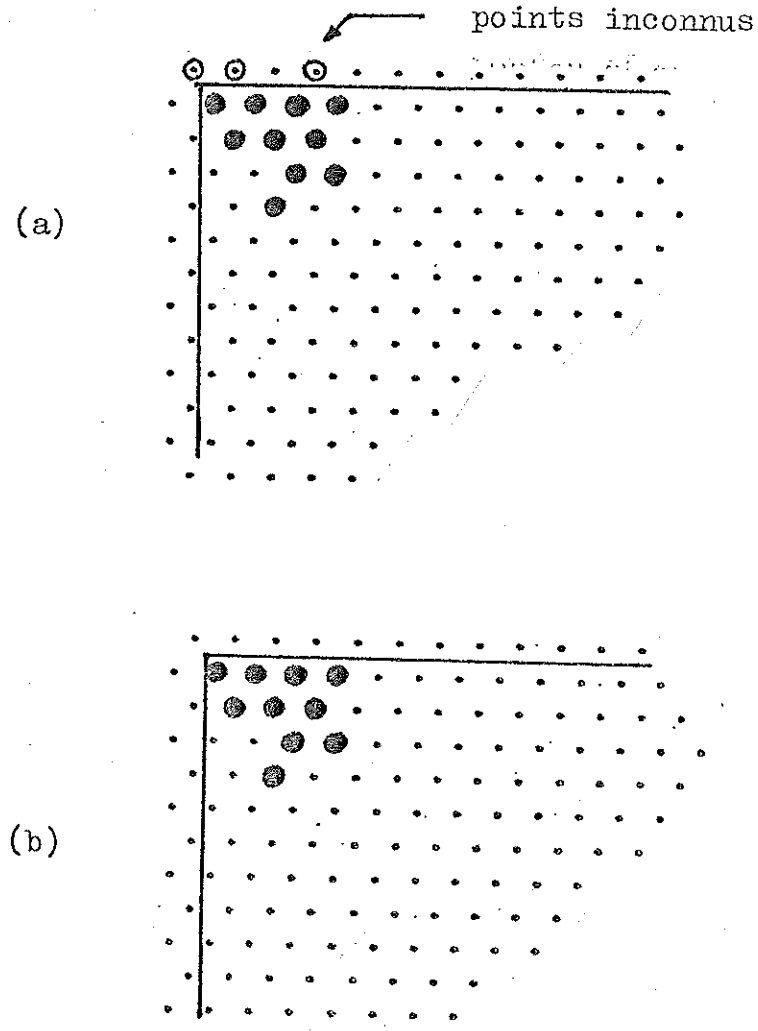


Figure 4 : Effets de bord

a) image coupant le bord du champ

b) par convention, les points extérieurs au champ sont à zéro

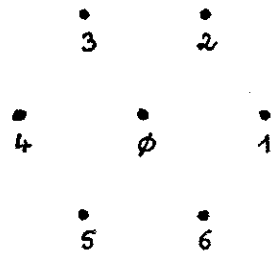
De la même façon, il faudra être très attentif aux itérations des transformations élémentaires (Dilater n fois un ensemble par un segment de taille 1 peut ne pas être équivalent à une dilatation de taille n).

Toutes ces particularités peuvent se résumer de la façon suivante : La convention adoptée entraine que les transformations effectuées sont des transformations géodésiques à l'intérieur d'un masque de taille 256 x 256 (pour plus de détails, voir [6]). Il faut néanmoins signaler que cette convention n'est pas plus mauvaise qu'une autre, lorsque aucune information n'est disponible à l'extérieur du masque.

Cette convention entraine également que le traitement par morceaux d'une image n'est pas équivalent au traitement en un seul passage (Par exemple, le traitement d'une image 512 x 512 par découpage en quatre images 256 x 256).

D/ Parallèlisation et accès au voisinage

D'après ce qui a été dit plus haut, le problème le plus important à résoudre lors de la programmation est de mettre en correspondance (afin d'effectuer une opération quelconque) un point et un point quelconque de son voisinage dans deux zones de la mémoire de travail. Examinons les différents cas possibles. Pour cela considérons l'hexagone suivant sur lequel les différents points ont été numérotés :



Puisque nous voulons effectuer une opération entre les deux points que nous mettons en correspondance, il est donc important si l'on veut utiliser au maximum la structure parallèle de PROPAL 2 que ces deux points soient résidents dans la mémoire

de travail d'un même processeur. En ce qui concerne les relations entre les points Ø-1 et Ø-4, cette règle est respectée ipso facto. Examinons plus en détail les autres correspondances. Deux cas peuvent se produire selon que le point central appartient à une ligne paire ou une ligne impaire. Les figures 5a .... 5c illustrent quelques situations typiques.

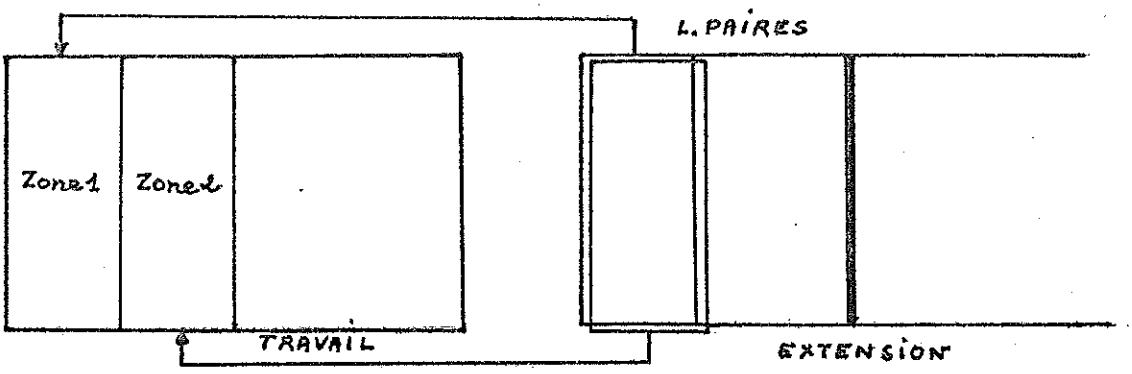
On remarquera, à l'examen de ces figures, que dans tous les cas, pour une certaine parité de la ligne les points sont en place, pour l'autre la mise en place se résume à un simple décalage vers le haut ou vers le bas de la zone de la mémoire de travail contenant le point du contour.

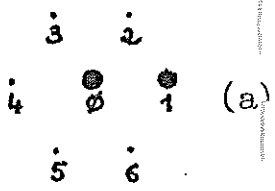
La séparation des lignes paires et impaires est donc assez performante, puisque pour la plupart des correspondances on limite les mouvements de l'ascenceur de PROPAL 2.

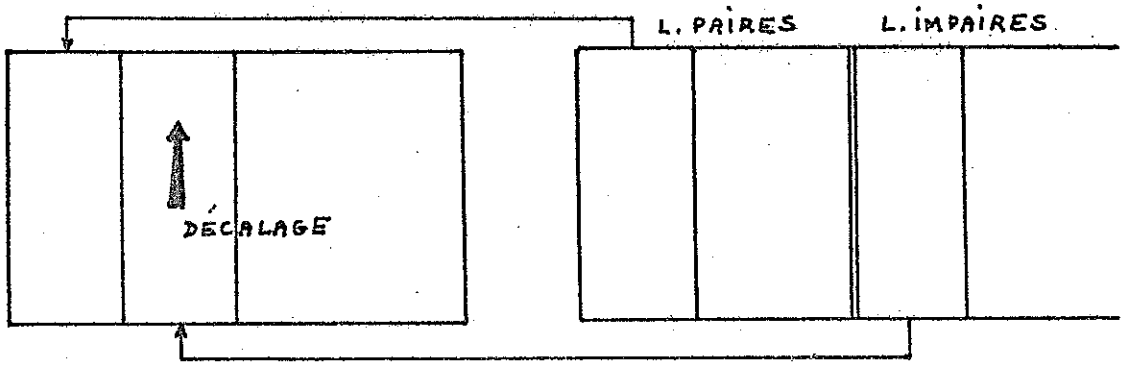
E/ Les transformations

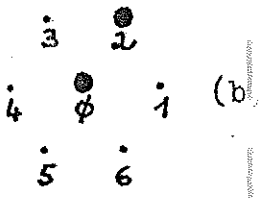
La palette de transformations élémentaires que nous avons choisie l'a été de façon à optimiser la souplesse d'utilisation sans pour autant augmenter de façon démesurée la taille de l'ensemble des programmes. Ainsi, outre le fait déjà signalé précédemment que seules des transformations de taille 1 ont été programmées, certaines d'entre elles ont été laissées de côté (comme par exemple l'ouverture, la fermeture).

Ces transformations ont été écrites en assembleur PROPAL. Etant donné la structure des mémoires et les contraintes imposées, la lecture de ces programmes est relativement complexe. Pour cette raison, on ne saurait trop recommander aux utilisateurs de ne pas les modifier inconsidérément. Nous avons classé ces transformations en quatre groupes : les transformations booléennes, les transformations morphologiques, les transformations utilitaires (tests, sélection du premier point, etc...) et enfin les mesures. Ces transformations seront décrites maintenant. Signalons que l'utilisateur pourra en trouver un résumé (syntaxe, procédure d'appel, etc...) dans le chapitre suivant.

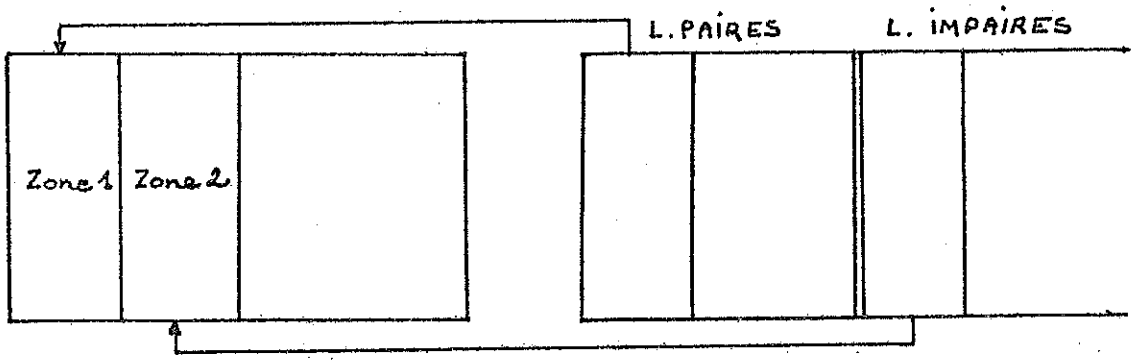


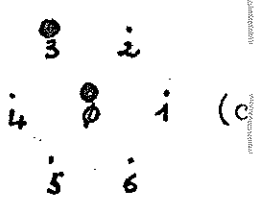






* Le point central appartient à une ligne paire





* Le point central appartient à une ligne impaire

Figure 5 : Exemples d'accès au voisinage

1/ Les opérations booléennes

Ces opérations sont effectuées par deux routines, l'une pour les opérations monadiques, l'autre pour les diadiques.

a) SGBOOL (SinGle BOOlean) : Opérations monadiques

Cette routine effectue à la demande, une des quatre opérations unaires suivantes :

- mise à $\emptyset$ d'une image
- mise à 1
- Transfert d'une mémoire à une autre
- Transfert avec complémentation d'une mémoire à une autre.

En fait l'unique transformation booléenne est la complémentation. Les autres peuvent être considérées comme des opérations d'initialisation.

b) DIBOOL (DIadic BOOlean) : opérations diadiques

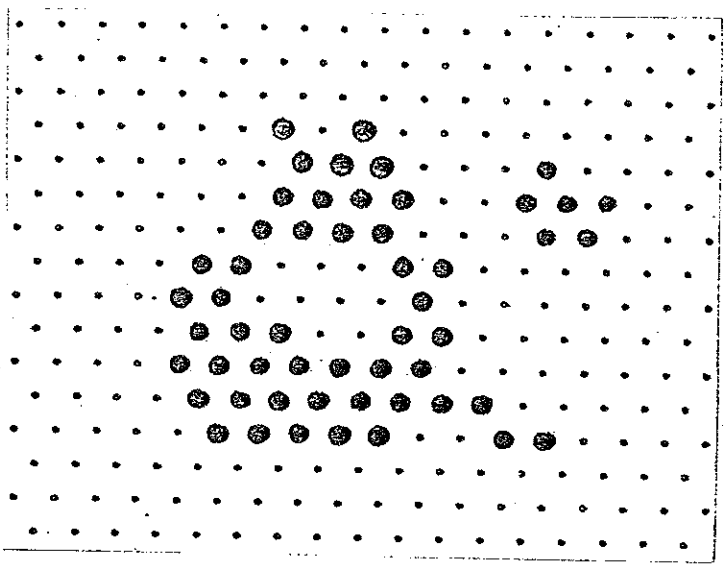
Le code PROPAL étant suffisamment riche en opérations logiques, cette routine peut effectuer un nombre assez important d'opérations diadiques. Ainsi, si IM1 et IM2 représentent les deux opérandes image, DIBOOL peut effectuer les opérations suivantes :

- intersection : $IM1 \cap IM2$
- intersection complémentée : $(IM1 \cap IM2)^c$
- différence ensembliste : $IM1/IM2$
- union : $IM1 \cup IM2$
- union complémentée : $(IM1 \cup IM2)^c$
- différence complémentée : $(IM1/IM2)^c$
- différence symétrique : $IM1 \Delta IM2$

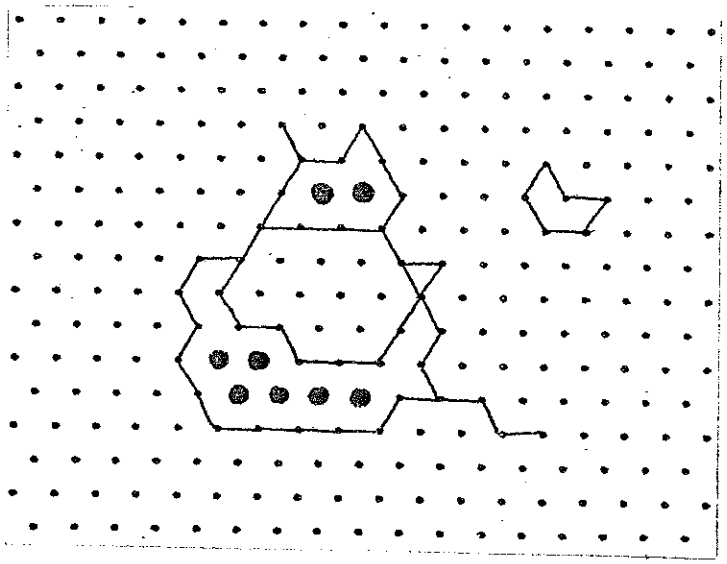
2) Les opérations morphologiques

a) ERODE : Erosion hexagonale

Cette routine effectue une érosion hexagonale de taille 1 d'une mémoire-image.

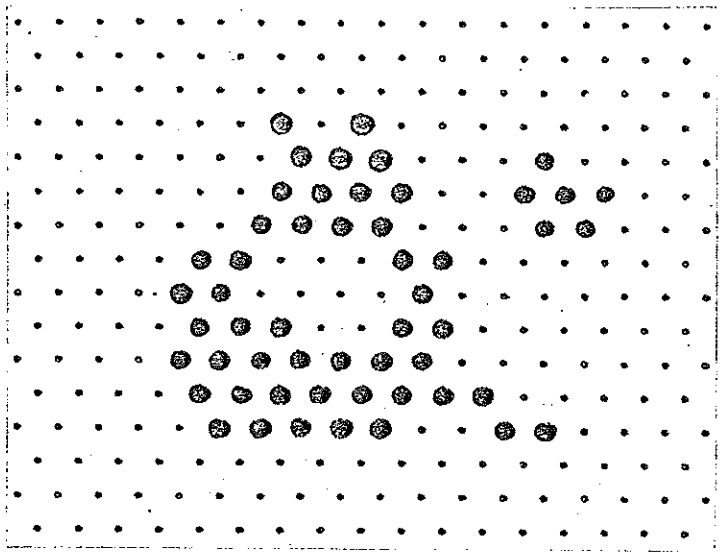


(a)

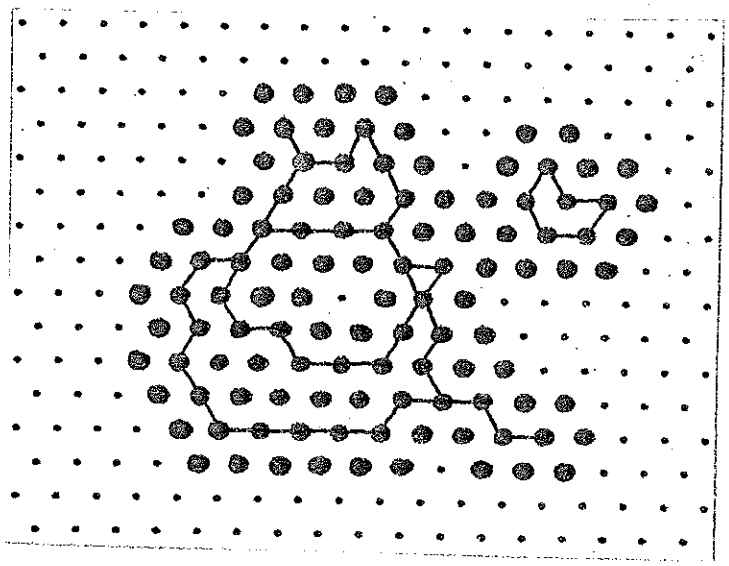


(b)

Figure 6 : Erosion hexagonale
a) initial
b) érodé



(a)



(b)

Figure 7 : Dilatation hexagonale
a) initial
b) dilaté

Soit X l'ensemble initial, et H l'hexagone élémentaire. L'érodé de X par l'élément structurant H est l'ensemble des points de l'image tels que H implanté en ces points soit inclus dans X (Fig. 6).

Il est facile de voir que cet érodé s'obtient en chaque point par l'intersection des 7 points de l'hexagone centré en ce point. L'érodé de X par H est noté $X \ominus H$. On montre également que :

$$X \ominus H = (((X \ominus S_1) \ominus S_2) \ominus S_3) \qquad \text{où } S_1, S_2$$

et S_3 sont les trois segments de droite élémentaires (doublets de points) définis dans les trois directions principales de la trame hexagonale (cf.[5]).

Cette formule a été utilisée pour programmer l'érosion. Elle présente l'intérêt de permettre de comparer les points deux à deux d'une part et de n'effectuer que 3 comparaisons (au lieu de 6) d'autre part.

b) DILATE : Dilatation hexagonale

C'est la transformation duale de l'érosion. On la note $X \oplus H$. L'ensemble obtenu est celui balayé par l'hexagone élémentaire quand son centre est astreint à rester dans l'ensemble initial X (Fig. 7).

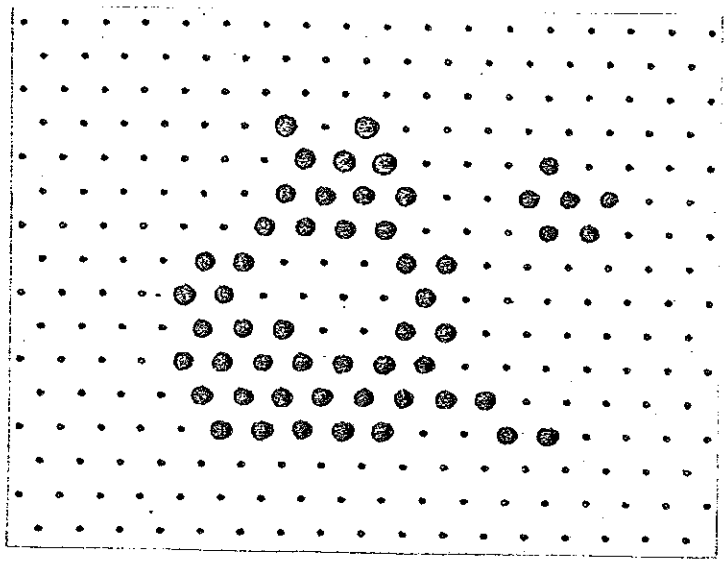
On montre que :

$$(X \oplus H) = (((X \oplus S_1) \oplus S_2) \oplus S_3)$$

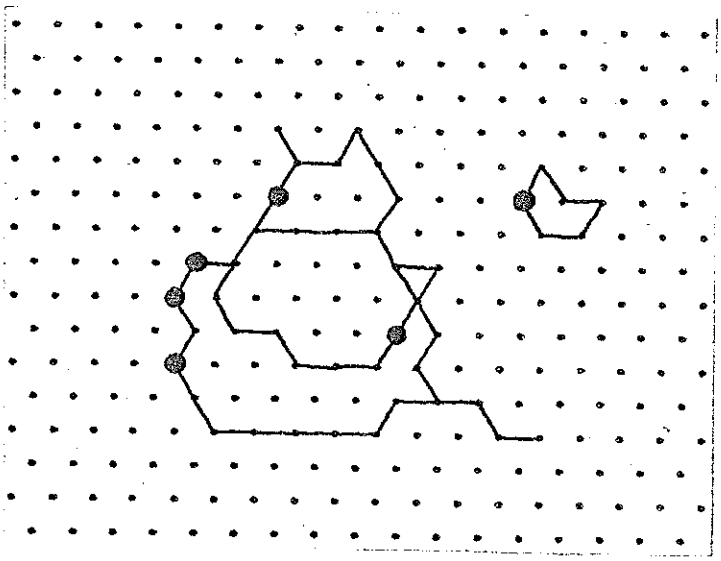
La programmation de cette routine suit le même schéma que celui de l'érosion.

c) LDC : Logique de Contour

Cette transformation est l'une des plus générales que l'on puisse concevoir pour les images binaires. En fait, on aurait très bien pu se contenter de programmer cette transformation. Les autres transformations morphologiques peuvent être ramenées à une LDC. En fait, si nous n'avons pas succombé à cet élan de paresse, c'est uniquement pour pouvoir améliorer les performances des opérations d'érosion et de dilatation.



(a)



(b)

Figure 8 : Transformée par tout ou rien (LDC)
a) initial
b) transformée

Soit X l'ensemble initial. Donnons nous sur l'hexagone élémentaire, une certaine configuration. La transformation logique de contour consiste à rechercher et marquer les points de l'image initiale dont le voisinage correspond à cette configuration. Cette opération est encore appelée transformation en tout ou rien (Fig. 8).

Nous désignons, sur l'hexagone élémentaire, par T_1 l'ensemble des points à 1 et par T_2 l'ensemble des points à $\emptyset$ (il n'est pas nécessaire que $T_1 \cup T_2 = H$; certains points peuvent très bien ne pas être pris en compte). Ce couple d'élément structurant est désigné par $T = (T_1, T_2)$. On peut montrer que cette transformation notée $X * T$ peut s'écrire :

$$X * T = (X \ominus T_1) \cap (X^c \ominus T_2)$$

d) THIN (THINning) ; THICK (THICKening) : amincissement et épaississement.

Ces deux transformations sont des extensions de la logique de contour. La première, l'amincissement consiste à supprimer de l'ensemble initial, les points mis en évidence par le LDC, la seconde au contraire consiste à ajouter ces points à l'ensemble initial. On peut donc écrire :

Amincissement : $X \circ T = X / (X * T)$
Epaississement : $X \odot T = X \cup (X * T)$

La nécessité, en Morphologie Mathématique, de faire de ces transformations des modules individualisés apparaît très souvent du fait de leur universalité. De plus, nous verrons lors de la description des transformations sur images de teintes que seules ces dernières transformations peuvent être définies, la transformation en tout ou rien n'ayant pas d'équivalent en termes de fonction. Il est évident que les éléments structurants $T = (T_1, T_2)$ doivent vérifier certaines propriétés pour que ces transformations soient efficaces. Le lecteur est prié de consulter la littérature sur le sujet ([5], [7]).

3) Les mesures

Nous avons déjà signalé que les mesures compatibles avec les principes fondamentaux de la Morphologie Mathématique étaient peu nombreuses. En fait, on peut montrer que ces mesures sont au nombre de trois dans le cas d'ensembles plans. Ces trois mesures sont :

- la surface
- le nombre d'intercepts
- le nombre de connexité

Avant de décrire ces mesures, il convient de dégager succintement le processus utilisé en morphologie mathématique pour effectuer une mesure. Ce processus est le suivant : L'ensemble à mesurer est transformé, et un comptage de points est effectué sur l'ensemble transformé (voir plus loin). Ceci implique que les routines de mesures font toutes appel à une routine commune, COUNT qui est une routine de comptage. De plus, du fait même de la nécessité d'un comptage, opération séquentielle par essence, il n'est plus possible dans PROPAL 2 de parallèliser complètement le processus. En conséquence, ces mesures seront sensiblement plus longues que les opérations morphologiques.

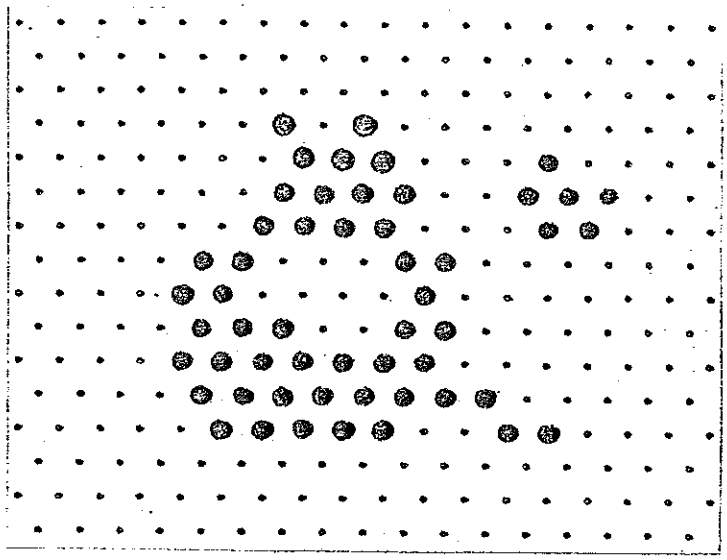
a) SURF (SURFace) : mesure de surface)

Cette routine est la plus simple, puisque la transformation est l'identité. SURF calcule le nombre de points blancs de l'image.

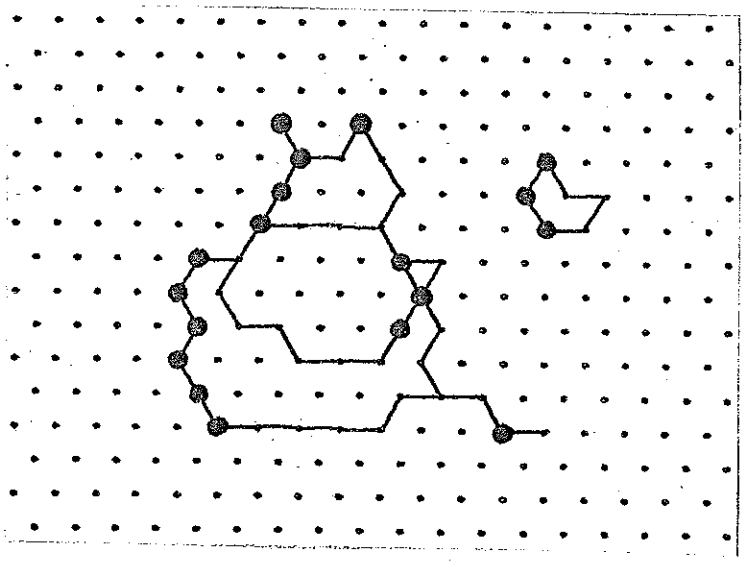
b) INTER1, INTER2, INTER3 : nombre d'intercepts

Ces routines calculent le nombre d'intercepts dans les 3 directions principales de la trame hexagonale.

Le nombre d'intercepts est le nombre d'entrées dans un ensemble. C'est donc le nombre de configurations $\emptyset 1$ (si on se place dans la direction 1) présentes dans cet ensemble. La transformation effectuée est donc ici une transformation en tout ou rien par le doublet $\emptyset 1$. COUNT est utilisé pour compter le nombre d'occurences obtenu (Fig. 9).

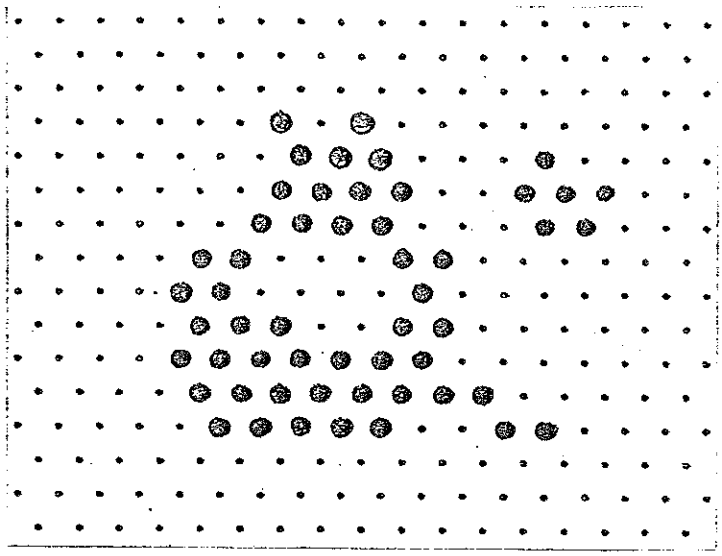


(a)

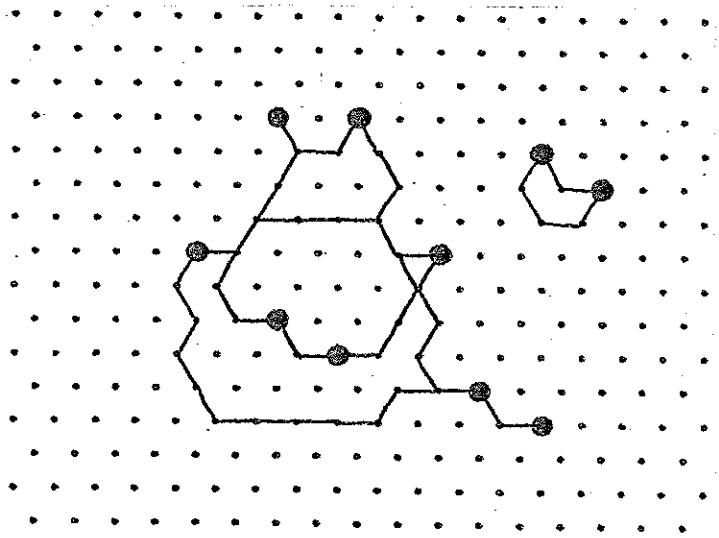


(b) I1 = 18

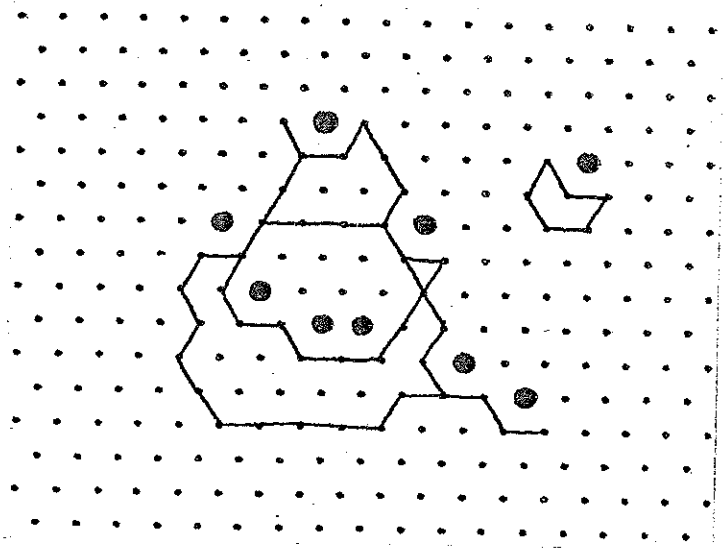
Figure 9 : Nombre d'intercepts
a) image initiale
b) transformation et comptage



(a)



(b) $\nu 1 = 1\emptyset$



(c) $\nu 2 = 9$

$\nu_c = \nu_1 - \nu_2 = 1$

Figure 10 : Nombre de connexité
a) initial
b) transformée n° 1 et comptage
c) transformée n° 2 et comptage

Ce nombre d'intercepts correspond à un facteur d'échelle près à la variation diamétrale dans l'espace continu.

Il est possible de calculer le périmètre d'un ensemble lorsque l'on connaît les nombres d'intercepts dans les directions principales de la trame. Il suffit pour cela d'utiliser une approximation digitale d'une formule de géométrie intégrale, la formule de Cauchy-Crofton. L'estimation du périmètre est la suivante :

$$P = \frac{\pi}{3} [I_1 + I_2 + I_3]$$

c) CONXT (CONneXiTé) : Nombre de connexité

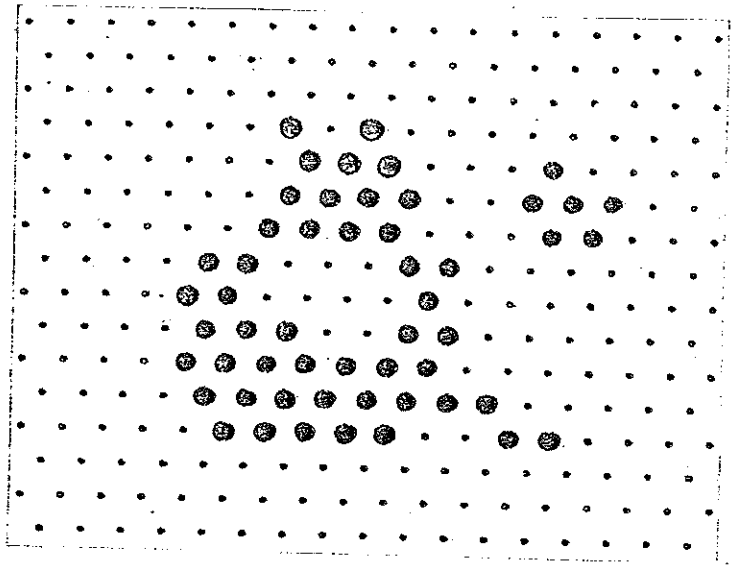
Dans R^2, le nombre de connexité d'un ensemble est égal au nombre de composantes connexes de cet ensemble diminué du nombre de trous. Des considérations théoriques trop longues à expliciter ici (cf. [5]) montrent que ce nombre est égal à la différence entre le nombre de configurations $\binom{0}{1}\,^{0}$ et le nombre de configurations $\binom{1}{1}\,^{0}$. La transformation effectuée lors de cette mesure est donc double puisqu'il faut effectuer deux transformations par tout ou rien à l'aide des deux éléments structurants précités (Fig. 1Ø).

4) Les utilitaires

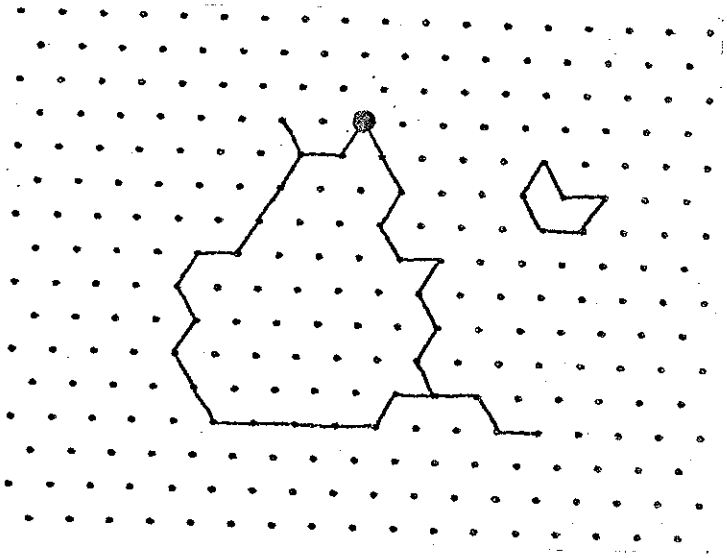
Cette rubrique regroupe un certain nombre de routines qu'il est relativement difficile de classer : routines de tests, d'entrée/sortie, etc...

a) PRPT (PRemier PoinT) : Récupération du "premier" point de l'image

Comme son nom l'indique, cette routine génère un ensemble réduit à un point, ce point était le premier point en haut à droite de l'ensemble initial. Cette routine permet en quelque sorte de marquer la première particule d'un ensemble, les particules étant ordonnées suivant un balayage descendant puis de droite à gauche (Fig. 11).



(a)



(b)

Figure 11 : Premier point

a) image initiale

b) extraction du premier point en haut puis à droite

Cette routine est fort utile dans les applications nécessitant une analyse individuelle des particules d'un ensemble.

b) IMTEST (IMage TEST) : Test d'identité de deux images

Cette routine positionne un indicateur permettant de savoir si les deux images considérées sont identiques ou non. Ce test est très important. En effet, la puissance des transformation morphologiques réside dans le fait de pouvoir les itérer. Or, qui dit itération, dit obligatoirement Test de sortie. La propriété utilisée en Morphologie pour sortir d'une itération est l'idempotence. Elle signifie qu'une itération supplémentaire d'une transformation ne change plus l'ensemble précédent. Il est donc fort utile de disposer d'un tel test d'idempotence. Ce test détecte bien l'identité de deux ensembles (c'est-à-dire le fait que $X_1 \Delta X_2 = \emptyset$) et non pas simplement l'égalité de leur surface ou même l'invariance de leur forme (deux ensembles translatés ont même forme mais ne sont pas identiques).

Ce test d'idempotence est le seul test implanté dans le logiciel binaire. L'expérience montre que dans presque tous les cas d'autres tests éventuels (tels que l'inclusion par exemple) ne présentent aucun intérêt.

c) DUMP : Sortie imprimante d'une portion d'image binaire

Cette routine imprime suivant une trame hexagonale une partie de l'image binaire spécifiée.

d) CHARGT (CHARGemenT) : Chargement d'une image

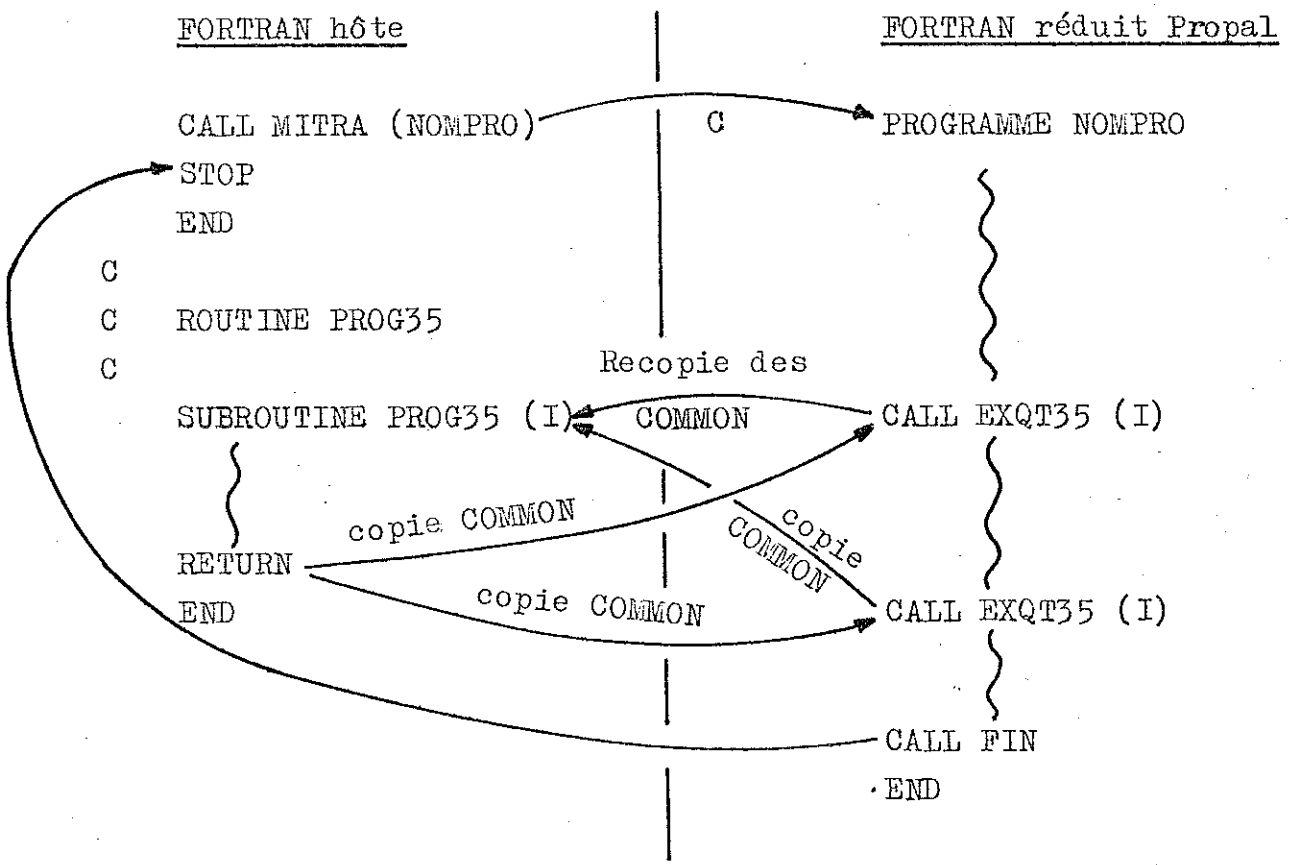
Cette routine permet le chargement en mémoire d'extension PROPAL d'une image présente dans la mémoire de l'Unité de Gestion.

F/ Mise en oeuvre des programmes

Les routines décrites précédemment sont appelées par un programme FORTRAN réduit implanté dans l'U.G.. Ce FORTRAN réduit ne manipule pas les données en flottant et les entrées/sorties standard du FORTRAN (comme READ, WRITE, etc...). L'utilisateur

devra avoir à l'esprit ces restrictions lorsqu'il élaborera un programme FORTRAN Propal. Néanmoins, pour pallier ces défauts, l'ADERSA-GERBIOS a développé un logiciel bi-tâche permettant l'interface logiciel entre l'ordinateur-hôte et Propal 2. Ce logiciel permet en particulier l'accès au FORTRAN standard. Le passage des données du FORTRAN-hôte au FORTRAN réduit s'effectue par l'intermédiaire de directives COMMON.

Le déroulement d'un programme est séquentiel : au démarrage, la tâche principale est lancée. Sa fonction est d'appeler la tâche principale Propal. Cette tâche Propal peut à tout instant faire appel à des routines du programme-hôte, par l'intermédiaire d'un appel standard : EXQT35 (i). Cet appel a pour effet d'exécuter la routine PROG35 (i) résidente dans le programme-hôte. Dès que cette routine a été exécutée, on retourne en séquence dans le programme FORTRAN Propal. La variable i permet d'aiguiller le programme sur plusieurs fonctions différentes. L'exemple ci-dessous illustre succintement l'ensemble des opérations. On pourra également consulter [4] pour de plus amples informations.



L'utilisateur trouvera dans les fiches-cuisine la syntaxe des différentes routines d'analyse des images.

Le programme ci-dessous illustre l'utilisation des différents sous-programmes. Ce programme effectue le squelette d'une image-test.

```
C      PROGRAMME DE SQUELETTE
C        IMAG1 : IMAGE A SQUELETTISER
C        IMAG2 : IMAGE SQUELETTE
C
         CALL  SGBOOL (IMAG1, IMAG2, 2)
         CALL  SGBOOL (IMAG2, ITEMP, 2)
         CALL  THIN   (IMAG2, IMAG2, 1, 1, 1, -1, 0, 0, -1)
         CALL  THIN   (IMAG2, IMAG2, 1, -1, 1, 1, -1, 0, 0)
         CALL  THIN   (IMAG2, IMAG2, 1, 0, -1, 1, 1, -1, 0)
         CALL  THIN   (IMAG2, IMAG2, 1, 0, 0, -1, 1, 1, -1)
         CALL  THIN   (IMAG2, IMAG2, 1, -1, 0, 0, -1, 1, 1)
         CALL  THIN   (IMAG2, IMAG2, 1, 1, -1, 0, 0, -1, 1)
         CALL  IMTEST (IMAG2, ITEMP, I)
         IF (I.EQ.Ø)  GOTO 2Ø
         STOP
         END
```

Les pages suivantes contiennent les descriptifs des différents sous-programmes élaborés, leur procédure d'appel. Certains exemples d'utilisation ont été ajoutés.

Nom : SGBOOL

But : Opération monadiques sur des images implantées en mémoire d'extension Propal 2 (image 256x256x1)

Appel Fortran :

CALL SGBOOL (IMAG1, IMAG2, ICODE)

Arguments :

IMAG1 : adresse mémoire M2 image initiale

IMAG2 : adresse mémoire M2 image résultante

ICODE : nature de l'opération

ICODE = Ø : mise à zéro image IMAG2
= 1 : mise à 1
= 2 : copie IMAG1 dans IMAG2
= 3 : complémentation

Remarque : Les images sont repérées par leur adresse de poids faible.
On peut faire IMAG2 = IMAG1, sans ambiguité.

Nom : DIBOOL

But : Opérations booléennes diadiques sur des images 256x256x1 implantées en mémoire d'extension.

Appel Fortran :

CALL DIBOOL (IMAG1, IMAG2, IMAG3, ICODE)

Arguments :

IMAG1 : adresse image opérande 1
IMAG2 : adresse image opérande 2
IMAG3 : adresse image résultante
ICODE : nature de l'opération

ICODE = Ø :	IMAG1 $\cap$ IMAG2	(And)
= 1 :	$\overline{\text{IMAG1} \cap \text{IMAG2}}$	(Nand)
= 2 :	IMAG1 $\cap$ $\overline{\text{IMAG2}}$	(Dif)
= 3 :	IMAG1 $\cup$ IMAG2	(Or)
= 4 :	$\overline{\text{IMAG1} \cup \text{IMAG2}}$	(Nor)
= 5 :	$\overline{\text{IMAG1}} \cup$ IMAG2	(Ndif)
= 6 :	IMAG1 Δ IMAG2	(Xor)

Remarque : Les images sont repérées par leur adresse de poids faible.
On peut faire IMAG1 = IMAG2 = IMAG3.

Nom : ERODE

But : Erosion hexagonale de taille 1 d'une image 256x256x1 implantée en mémoire d'extension

Appel Fortran

```
CALL ERODE (IMAG1, IMAG2)
```

Arguments :

IMAG1 : adresse M2 image initiale
IMAG2 : adresse M2 image finale

Remarque : Les images sont repérées par leur adresse de faible poids.
On peut faire IMAG1 = IMAG2.

Exemple :

Erosion de taille I de IMAG1, résultat dans IMAG2 :

```
      IMAG = IMAG1
      DO 5  NUM = 1,I
      CALL  ERODE (IMAG, IMAG2)
      IMAG  = IMAG2
5     CONTINUE
```

Nom : DILATE

But : Dilatation par un hexagone de taille 1 d'une image 256x256x1 implantée en mémoire d'extension.

Appel Fortran :

CALL DILATE (IMAG1, IMAG2)

Arguments :

IMAG1 : adresse mémoire initiale
IMAG2 : adresse mémoire finale

Remarque :

Voir fiche-cuisine Erosion

Exemple : Ouverture de taille I de IMAG1, résultat dans IMAG2.:

```
      IMAG = IMAG1
      DO 5  NUM = 1, I
      CALL  ERODE (IMAG,IMAG2)
      IMAG = IMAG2
5     CONTINUE
      DO  1Ø  NUM = 1, I
      CALL  DILATE (IMAG2, IMAG2)
1Ø    CONTINUE
```

Nom : LDC

But : Effectue une opération de logique de contour hexagonal de taille 1 sur une image binaire 256x256 implantée en mémoire d'extension.

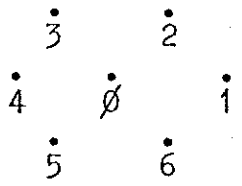
Appel Fortran :

CALL LDC (IMAG1, IMAG2, IØ, I1, I2, I3, I4, I5, I6)

Arguments :

IMAG1 : adresse image initiale
IMAG2 : adresse image résultante
IØ : valeur prise par le point central
I1,...,I6 : Valeurs prises par les points du contour

Remarque : Le numérotage des points du contour est le suivant :



Lorsqu'un point n'est pas pris en compte (il peut prendre alors indifféremment la valeur 0 ou 1), son argument prend la valeur -1.

Exemple :

Translation vers la droite d'une image (I pas)

```
      DO  1ØØ  NUM = 1, I
      CALL LDC (IMAG, IMAG, -1, -1, -1, -1, 1, -1, -1)
1ØØ   CONTINUE
```

Nom : THIN

But : Amincissement d'une image binaire implantée en mémoire d'extension.

Appel Fortran :

CALL THIN (IMAG1, IMAG2, IØ, I1, I2, I3, I4, I5, I6)

Arguments :

Leur signification est la même que pour LDC.

Exemple :

Programme de squelette (voir texte)

Nom : THICK

But : Epaississement d'une image binaire implantée en mémoire d'extension

Appel Fortran :

CALL THICK (IMAG1, IMAG2, IØ, I1, I2, I3, I4, I5, I6)

Arguments :

Même signification que LDC et THIN

Nom : PRPT

But : "Premier point" d'une image. Génération d'une image contenant le premier point en haut et à droite de l'image initiale.

Appel Fortran :

```
CALL PRPT (IMAG1, IMAG2, IFLAG)
```

Arguments :

IMAG1 : Adresse de l'image initiale
IMAG2 : Adresse de l'image contenant le premier point
IFLAG : Indicateur en retour

IFLAG = 1 : un point à 1 a été trouvé
= Ø : Aucun point n'a été trouvé

Exemple :

Analyse individuelle de particules contenues dans IMAG1. IMAG2 tiendra la particule utilisée :

```
1Ø      CALL  PRPT (IMAG1, IMAG2, IFLAG)
        IF (IFLAG.EQ.Ø) GOTO 2Ø
        CALL RECONS (IMAG1, IMAG2)
C       RECONS EST UNE ROUTINE DE RECONSTRUCTION
C       D'ENSEMBLE VOIR EXEMPLE <IMTEST>
              insérer ici le programme
              d'analyse de la particule
              stockée dans IMAG2
        CALL DIBOOL (IMAG1, IMAG2, IMAG1, 2)
        GOTO  1Ø
2Ø      CONTINUE
```

Nom : IMTEST

But : Test d'identité de deux images implantées en mémoire d'extension.

Appel Fortran :

CALL IMTEST (IMAG1, IMAG2, ITEST)

Arguments :

IMAG1 : adresse mémoire première image
IMAG2 : adresse mémoire deuxième image
ITEST : Indicateur en retour :

ITEST = Ø images non identiques
ITEST = 1 images identiques

Exemple :

Test d'idempotence d'une transformation de reconstitution de particules

```
      SUBROUTINE  RECONS (IMAGE, MARKER)
      CALL  DIBOOL (IMAGE, MARKER, MARKER, Ø)
1Ø    CALL  SGBOOL (MARKER, ITEMP, 2)
      CALL  DILATE (ITEMP, MARKER)
      CALL  DIBOOL (IMAGE, MARKER, MARKER, Ø)
      CALL  IMTEST (MARKER, ITEMP, I)
      IF (I.EQ.Ø) GOTO 1Ø
      RETURN
      END
```

Nom : SURF

But : Mesure de l'aire (nombre de points) d'une image binaire en mémoire extension

Appel Fortran :

```
CALL SURF (IMAG, IVAL)
```

Arguments :

IMAG : adresse mémoire image

IVAL : Valeur de la surface en sortie

Nom : CONXT

But : Calcul du nombre de connexité sur une image binaire

Appel Fortran

```
CALL  CONXT (IMAG, IVAL)
```

Arguments

IMAG : adresse mémoire image

IVAL : En retour, contient la valeur du nombre de connexité.

Nom : INTER 1, INTER 2, INTER 3

But : Calcul du nombre d'intercepts dans les trois directions principales de la trame hexagonale sur image binaire.

Appel Fortran :

```
CALL INTER1 (IMAG, IVAL)
```

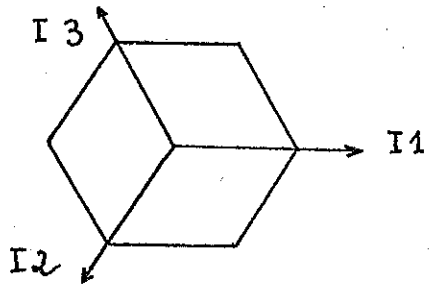
(idem pour INTER2, INTER3)

Arguments :

IMAG : adresse image à traiter

IVAL : valeur du nombre d'intercepts dans la direction choisie.

Remarque : Les trois directions principales sont les suivantes



Exemple :

Calcul du périmètre d'un ensemble en mémoire IMAGE par la formule de Crofton :

```
PI = 3.14159
CALL INTER1 (IMAGE, I1)
CALL INTER2 (IMAGE, I2)
CALL INTER3 (IMAGE, I3)
PERIM = PI * (I1 + I2 + I3)/3
```

(Note : Ce programme ne tourne pas en Fortran Propal réduit. Ce Fortran n'accepte pas les variables flottantes).

Nom : CHARGT

But : Chargement d'une image 256x256x1 bit, résidente en mémoire UG, dans la mémoire extension du Propal 2.

Appel Fortran :

CALL CHARGT (IMAG)

Arguments :

IMAG : adresse d'implantation en mémoire d'extension

Remarque : Ce programme est un programme provisoire susceptible de subir des modifications liées à l'implantation du logiciel numérique.

Nom : DUMP

But : Sortie sur imprimante en maillage hexagonal d'une bande de largeur 64 pixels d'une image binaire en mémoire d'extension.

Appel Fortran :

CALL DUMP(IMAG, ILIGN, ICOL, NLIGN)

Arguments :

IMAG : adresse mémoire d'image

ILIGN : Numéro de la ligne de début d'impression
($\emptyset \leq$ ILIGN ≤ 255)

ICOL : n° colonne de début d'impression
($\emptyset \leq$ ICOL ≤ 255)

NLIGN : Nombre de lignes dans la bande
($1 \leq$ NLIGN ≤ 256)

Remarque : Le programme ne vérifie pas la validité et la cohérence des paramètres ILIGN, ICOL, NLIGN

G/ Tests et Performances

Tous les tests de vitesse ont été effectués sur la même image-test. La complexité de l'image n'intervient pas dans la vitesse d'exécution des transformations de base. Ces vitesses ont été comparées à celles obtenues sur un analyseur de texture classique (l'analyseur LEITZ-TAS en l'occurence). Quelques programmes itératifs (squelette, reconstitution de particules) ont été écrits, afin d'être utilisés comme "bench marks" sur les deux appareils. Le tableau ci-après résume les diverses performances des deux appareils. On remarquera que les vitesses d'exécution des opérations de LDC ou d'amincissement/épaississement dépendent sur Propal de l'élément structurant utilisé. Les mesures sur Propal sont relativement longues par rapport aux transformations. La raison en a été donnée plus haut.

En résumé, le gain de temps lors d'opérations élémentaires est dans un facteur 6/7 par rapport à un analyseur classique. Il est de 8/9 lors d'itérations. Il convient de noter que la structure d'un programme est différente sur les deux machines. En particulier, la souplesse d'utilisation des mémoires d'images est plus grande avec Propal 2.

Transformation	PROPAL 2	Leitz-TAS (1)	Remarques
Mise à ∅ mémoire	0,64	20,0	
Mise à 1 mémoire	0,75	20,0	
Transfert mem1-mem2	0,81	20,0	
Transfert complémenté	0,91	20,0	
Opérations booléennes	1,53	20,0	(2)
Erosion de taille 1	3,40	20,0	
Dilatation de taille 1	3,43	20,0	
Ouverture/Fermeture	6,90	20,0	(3)
Transf. en tout ou rien	3,2 - 8,0	20,0	(4)
Epaississement/Amincis.	- id -	20,0	
Premier point	- id -	40,0	
Test égalité images	2,40 max	20,0 min	(5)
Mesure surface	20,43	20,0	
Mesure intercepts	20,78 - 21,22	20,0	(6)
Mesure connexité	42,62	20,0	
Test squelette	314,0	3000,0	
Test reconstruction	<2 s.	10 s.	

TABLEAU DE COMPARAISON DES VITESSES
D'EXECUTION DE PROPAL2 ET DU LEITZ-TAS

(1) les vitesses correspondent aux programmes compilés en FORTRAN

(2) les opérations booléennes sont l'union, l'intersection, la différence, la différence systématique, etc...

(3) Ces transformations s'obtiennent par combinaison d'une érosion et d'une dilatation sur PROPAL 2

(4) La vitesse de cette transformation dépend de la complexité de l'élément structurant sur PROPAL

(5) Sur le TAS, ce test d'égalité passe par une mesure de surface

(6) La mesure dans la direction 1 est plus rapide sur PROPAL

CONCLUSIONS PROVISOIRES

Certains points n'ont pas été abordés dans ce rapport. Les opérations d'entrée-sortie des images n'ont, par exemple, pas été décrites. Ces opérations dépendent du périphérique utilisé. L'utilisation d'une console de visualisation est envisagée. Les problèmes de passage d'une trame carrée à une trame hexagonale et inversement sont pour la plupart résolus. Les transformations d'interface entre les images numériques et les images binaires sont en cours d'écriture (il s'agit essentiellement de l'opération de seuillage d'une image). Ces opérations seront décrites dans le prochain rapport.

Parallèlement à l'élaboration de ces logiciels, nous avons envisagé de définir un langage de programmation permettant l'utilisation aisée des routines de transformation d'images. Le lecteur a sans doute remarqué que l'utilisation intensive d'instructions CALL, ainsi que la dénomination des images par leur adresse d'implantation ne contribuent pas à la souplesse d'écriture des programmes. Pour cette raison, il semble intéressant de définir un sur-ensemble du langage FORTRAN comprenant des instructions de traitement d'images. Ce sur-ensemble (appelons-le MORPHTRAN !) serait également capable de gérer la mémoire-image de Propal 2. L'utilisateur pourrait alors manipuler ses images, en les désignant par leur nom, sans se préoccuper de leur implantation réelle en mémoire. Ce langage est actuellement en cours de simulation. Pour cela, un traducteur MORPHTRAN → FORTRAN est à l'étude.

Les premiers résultats obtenus montrent qu'un processeur vectoriel est assez bien adapté aux traitements morphologiques. Les gains de vitesse par rapport aux analyseurs classiques sont intéressants. D'autre part, l'écriture d'un logiciel pour images de teinte est facilitée par la structure des mémoires que nous avons définie.

ANNEXE A : Descriptif détaillé des principales transformations binaires.

Cette rubrique recouvre la description détaillée des deux principales transformations binaires : l'érosion et la logique de contour. Les autres transformations utilisent des structures de mémoire plus simples ou similaires. La dilatation par exemple est effectuée sur le même principe que l'érosion. Il suffit simplement de modifier les transformations ensemblistes et les initialisations (de même que l'épaississement et l'amincissement par rapport à la transformation LDC).

L'érosion

Pour l'érosion, l'image est découpée en quatre parties. La transformation est effectuée totalement sur une partie de l'image avant de passer à la section suivante.

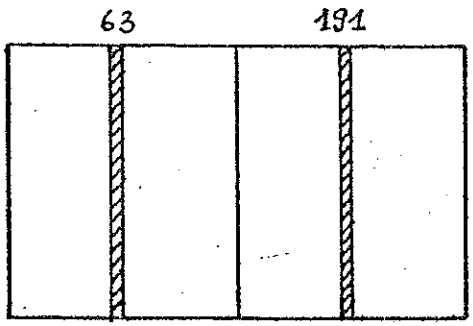
L'état de la mémoire de travail sera décrit à chaque étape. Les directions traitées sont indiquées par des flèches. Les lettres I et P désignent respectivement les lignes impaires et paires du bloc concerné.

La transformation en tout ou rien

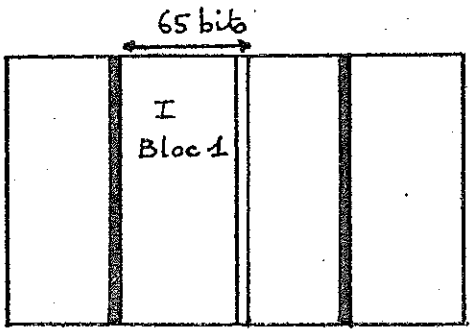
L'image est là encore traitée en quatre parties. Seules les directions programmées sont traitées. On remarquera le rôle de mémoire temporaire de stockage joué par la mémoire de travail (de même, d'ailleurs, que pour l'érosion).

SYNOPTIQUE DETAILLE DE L'EROSION HEXAGONALE

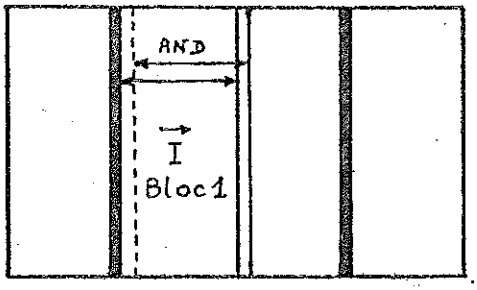
1) initialisation
RAZ des colonnes 63 et 191



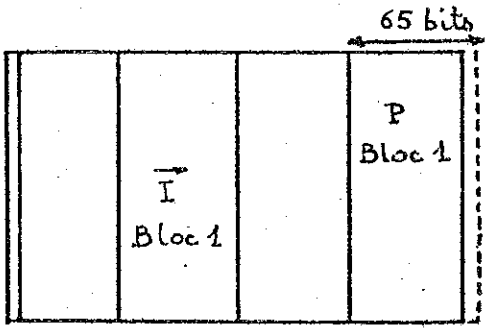
2) Chargement des lignes impaires du bloc 1 sur 65 bits (de 64 à 128)



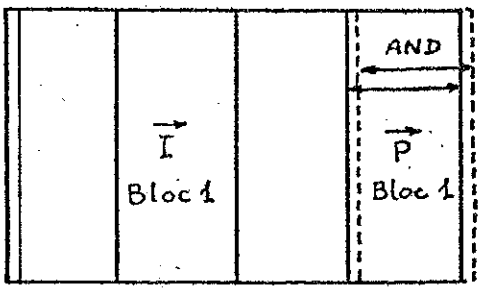
3) Opération ET sur les deux zones. Obtention de l'érodé par un segment dans la première direction (lignes impaires)



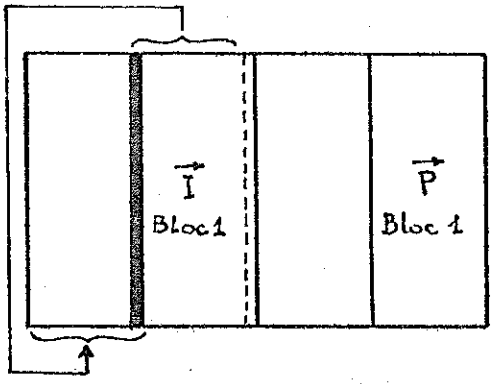
4) Chargement des lignes paires du bloc 1 sur 65 bits (de 192 à Ø avec "wrap around" de la mémoire)



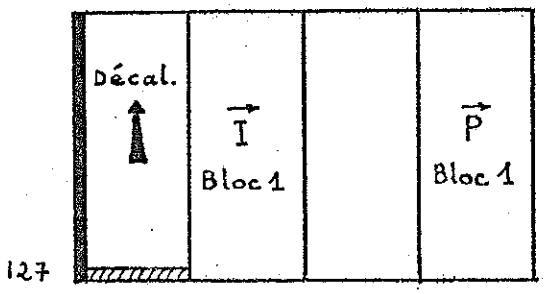
5) Opération ET et obtention de l'érodé dans la première direction (lignes paires)



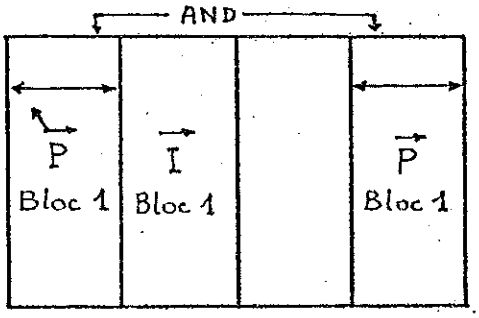
6) Transfert de la zone (63,126) dans (0,63)



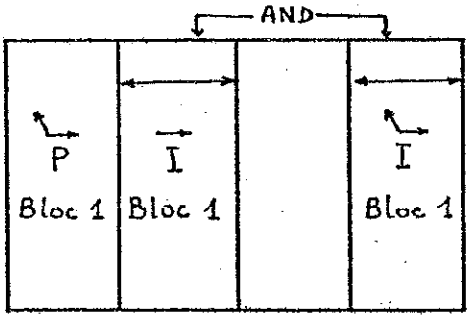
7) Décalage vers le haut de la zone (0,63) et mise à zéro de la ligne 127



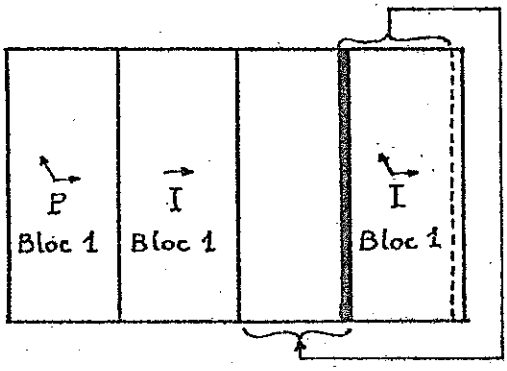
8) Opération ET entre les zones (192,255) et (0,63). Obtention de l'érodé dans les directions 1 et 2 pour les lignes paires.



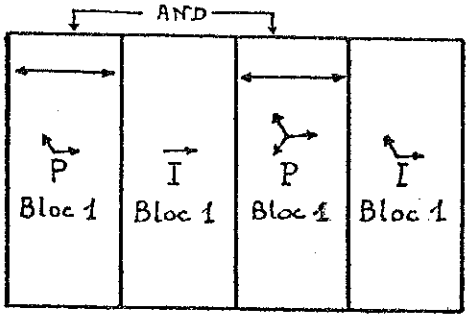
9) Opération ET entre les zones (192,255) et (64,127). Obtention de l'érodé dans les directions 1 et 2 pour les lignes impaires. (Résultat dans (192,255))



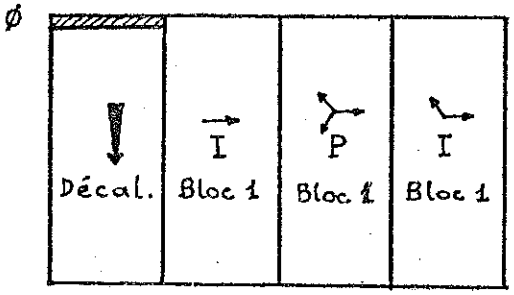
10) Transfert de la zone (181,254) dans (128,191)



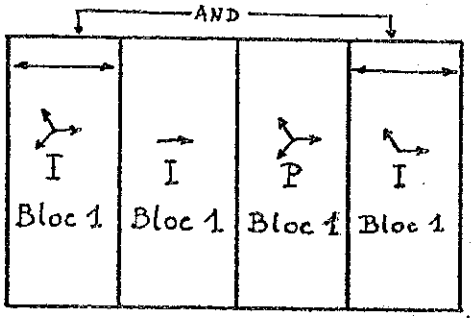
11) Opération ET entre les zones (0,63) et (128,191). Obtention de l'érodé hexagonal final du bloc 1 pour les lignes paires dans la zone (128,191)



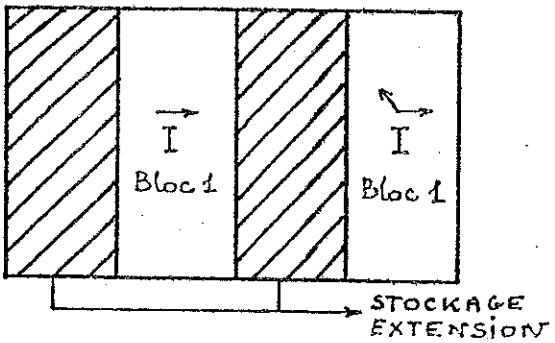
12) Décalage vers le bas de la zone (0,63) et mise à zéro de la ligne Ø.



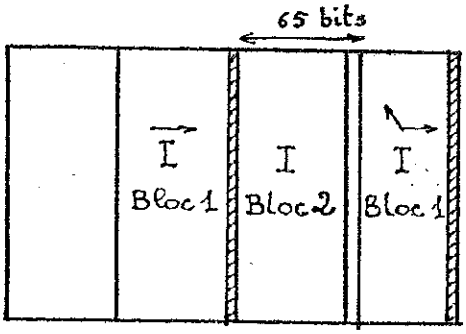
13) Opération ET entre les zones (0,63) et (192,255). Obtention de l'érodé hexagonal final du bloc 1 pour les lignes impaires



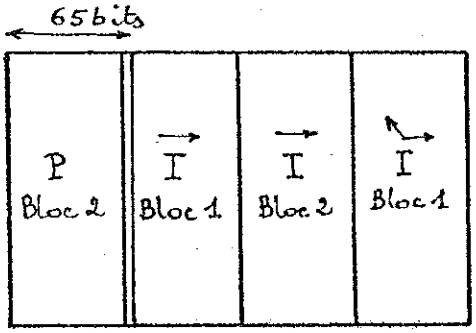
14) Stockage dans l'extension de la transformée pour le bloc 1



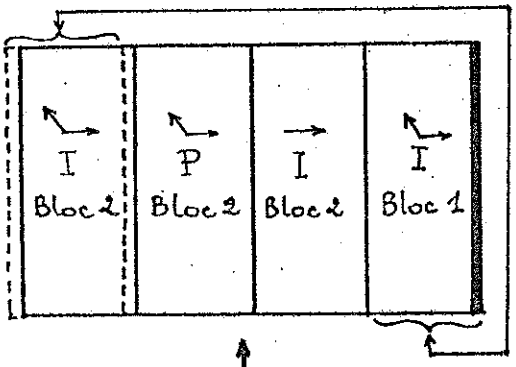
15) Chargement du bloc 2 des lignes impaires. Les transformées partielles impaires restant en mémoire vont propager l'effet de bord par l'intermédiaire de leur dernière colonne.



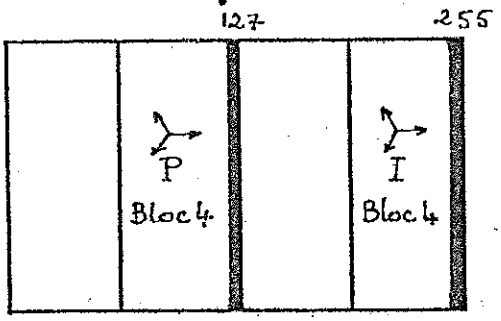
16) Les transformations se poursuivent normalement. Situation intermédiaire.



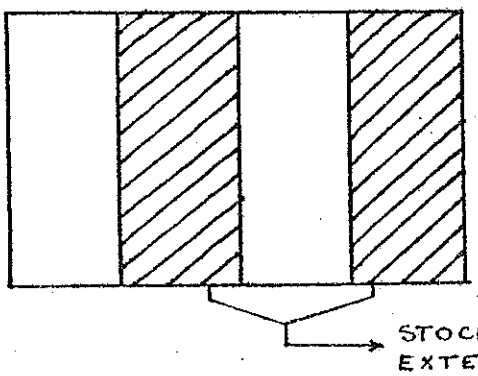
17) Propagation des transformations intermédiaires (suite)



18) Phase finale
mise à zéro des colonnes 127 et 255

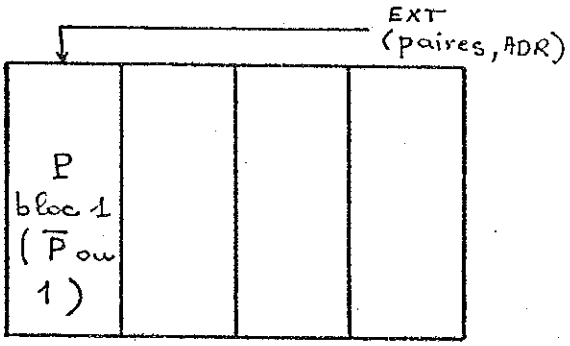


19) Stockage extension.
Fin de traitement

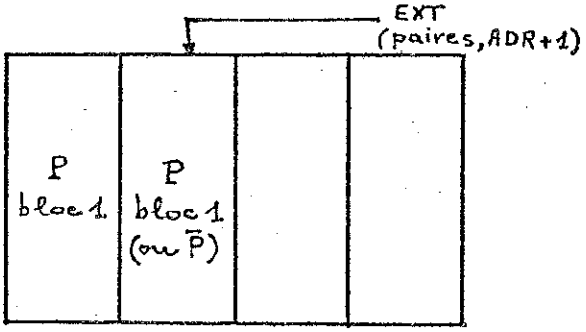


SYNOPTIQUE DETAILLE DE LA TRANSFORMEE EN TOUT OU RIEN (LDC)

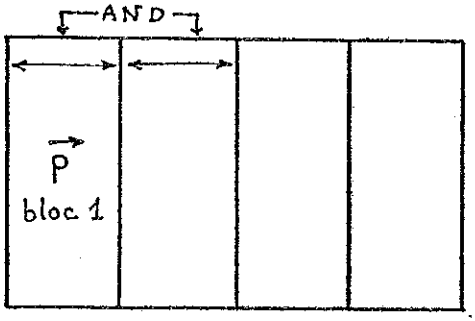
1) Traitement du point central chargement du bloc 1 pair (situé à ADR dans la mémoire d'extension) dans (0,63). Complémentation si point central à zéro



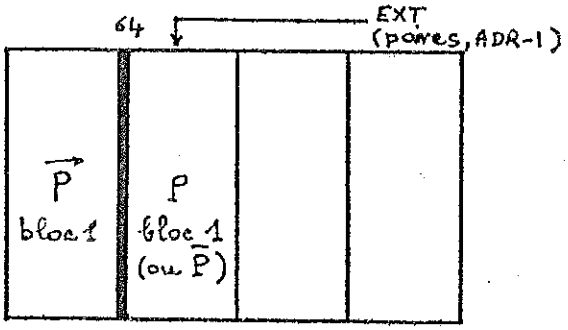
2) Traitement direction 1



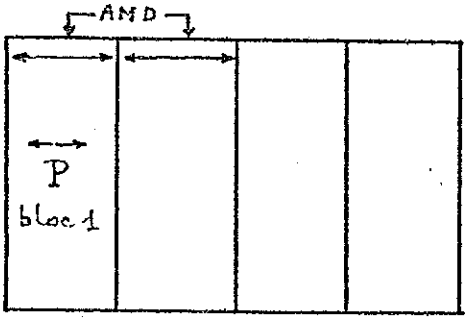
3) Opération booléenne (ET logique) résultat dans la zone (0,63)



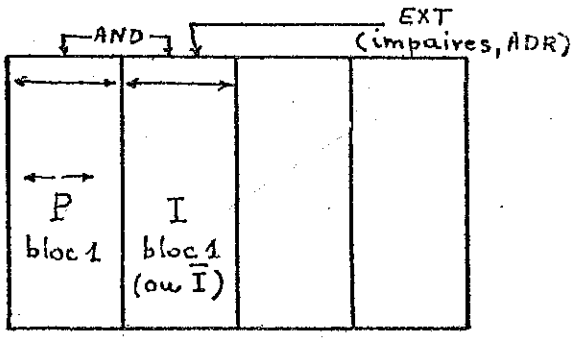
4) Traitement direction 4 (un RAZ de la colonne 64 est effectué pour le bloc 1)



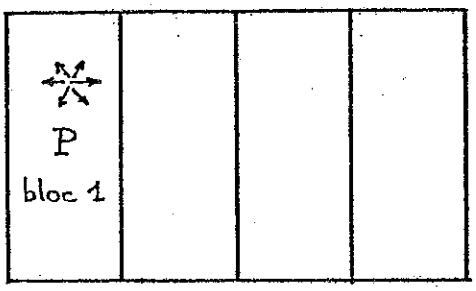
5) Opération ET résultat dans (0,63)



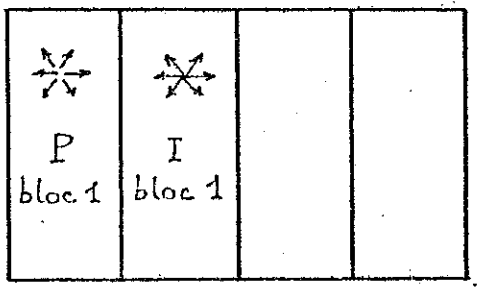
6) Traitement de la direction 6 et opération



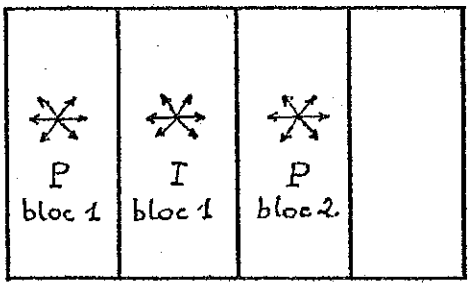
7) Les opérations se poursuivent
Le résultat final pour le bloc 1 (lignes paires) apparaît en (0,63)



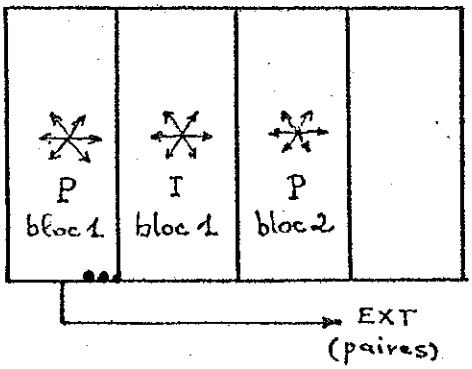
8) Etat de la mémoire de travail après traitement des lignes impaires



9) Bloc 2, traitement des lignes paires



10) stockage dans l'extension du bloc 1 (lignes paires) avant poursuite du traitement



etc...

Références bibliographiques

[1] - Iterative algorithms, H. DIGABEL et Ch. LANTUEJOUL, Proceedings of the Second European ISS Symposium, Caen, 1978.

[2] - PROPAL 2 : Manuel de Référence, CIMSA, Doc. 8215/U/FR, Janvier 1980.

[3] - Processeur parallèle associatif PROPAL 2 : présentation, CIMSA, Doc 8014/P3/FR, Décembre 79

[4] - PROPAL 2 : Manuel d'exploitation, CIMSA, Doc 8216/U/FR, Janvier 1980.

[5] - Image Analysis and Mathematical Morphology, J. SERRA, Academic Press, 1982.

[6] - On the use of geodesic metric in image analysis, S. BEUCHER et Ch. LANTUEJOUL, Journal of Microscopy, Vol. 121, Part 1, Janvier 1981

[7] - Shapes and patterns of microstructures considered as grey-tone functions, S. BEUCHER et J. SERRA, Proceedings 3rd European I.S.S. Symposium, Ljubljana, 1981.