MICROMORPH®

en

Questions

Serge BEUCHER, Dimitri GOROKHOVIK, Jean SERRA

Centre de Morphologie Mathématique

Ecole Nationale Supérieure des Mines de Paris

MICROMORPH® en questions

Comment MICROMORPH® se présente-t-il ?

L'environnement MICROMORPH® de démarrage est défini par le biais de deux fichiers: ***mmorph.ini*** et ***init.mic***. Nous vous suggérons fortement de ne pas modifier ces deux fichiers avant d'en avoir compris les fonctionnalités. La description détaillée de ces fichiers est fournie dans l'aide en ligne. En cas de modification de ***mmorph.ini*** vous pouvez toujours revenir à la situation antérieure en copiant ce fichier du CD-Rom vers le répertoire MICROMORPH®.

La machine de traitement d'images virtuelles constituant le coeur de MICROMORPH® est configurée pour contenir au démarrage 10 mémoires-images numériques 8 bits (intitulées **g1** à **g10**), 10 mémoires-images binaires (**b1** à **b10**) et 3 mémoires-images numériques 16 bits (**h1** à **h3**). Les procédures et fonctions de MICROMORPH® sont stockées dans un dictionnaire dont la taille est déterminée par ***mmorph.ini*** (elle devrait être largement suffisante pour la plupart de vos applications. Si ce n'était pas le cas, elle peut être modifiée ainsi que la plupart des caractéristiques de la machine virtuelle).

Comment démarrer ?

A la fin du lancement de MICROMORPH® s'ouvre d'une part la fenêtre de l'interpréteur de commandes, et de l'autre quatre fenêtres de visualisation. Comment s'en servir?

L'interpréteur de commande

La fenêtre de l'interpréteur de commandes permet de rentrer au clavier des commandes MICROMORPH® et de lancer des procédures. Cette fenêtre est formée de deux parties : la partie basse correspond à la ligne de commande courante, tandis que la partie haute contient l'historique des commandes déjà entrées. Vous pouvez facilement faire passer une ligne de commande de la partie haute à la partie basse (active) en la cliquant avec la souris (**Attention!** Pour entrer des commandes, il est indispensable que la fenêtre de l'interpréteur soit active; la couleur du bandeau au sommet de la fenêtre indique généralement son état. Pour activer une fenêtre non active, il suffit de cliquer dessus).

D'autre part, cette fenêtre peut être éditée grâce à la fonction ***edit*** de sa barre de menu. Enfin, la fonction ***file*** permet de compiler des procédures écrites à l'aide du langage de programmation et contenues dans un fichier texte (voir ci-dessous).

La visualisation

Vous pouvez visualiser plusieurs images en même temps dans MICROMORPH®. Au démarrage, quatre fenêtres de visualisation (2 binaires et 2 numériques) s'ouvrent et affichent les contenus des mémoires **g1**, **g2**, **b1** et **b2,** mais on peut en appeler d'autres. La gestion des fenêtres de visualisation peut se faire par le biais du sous-menu **windows** dans le menu principal. La visualisation peut se faire également à l'aide de commandes MICROMORPH®, comme **imdisplay**, **ds**, **dscol**, etc.

Pour charger une image dans une mémoire, vous pouvez passer par la commande MICROMORPH® **imload**. On peut aussi agir directement avec la souris. Il suffit pour cela, après avoir activé la fenêtre de visualisation correspondante en cliquant sur son bandeau, de cliquer sur la fonction ***file*** du menu principal pour l'activer, et enfin sur la fonction ***load***. Apparaissent alors les répertoires image où l'on choisit l'image à charger dans la mémoire séléctionnée. L'opération inverse (stocker un résultat de traitement dans un fichier-image par exemple) s'exécute par un cheminement analogue, où la seule différence est que ***save*** remplace ***load***.

Enfin l'image visualisée dans la fenêtre de visualisation active peut être chargée dans le presse-papier. Il suffit pour cela d'activer la fonction ***edit*** du menu principal. L'opération inverse est également faisable. Cette facilité permet donc d'échanger très rapidement des images avec d'autres applications (traitements de texte, logiciels de retouche d'images par exemple).

Quelles données MICROMORPH® traite-t-il ?

L'interpréteur du langage de MICROMORPH® exécute des opérations à partir des types des données suivants:

- entiers**,**
- chaînes de caractères,
- tableaux d'entiers,
- images.

Dans ce langage:

- Les calculs en virgule flottante ne sont pas supportés (en morphologie mathématique, ce type de données est généralement inutile et il ralentit les opérations).
- Les entiers ont 32 bits de profondeur. Ils sont bien supportés par les opérations arithmétiques. On peut avoir un entier dans la portée globale ou locale d'une fonction, ou dans le dictionnaire.
- Les chaînes de caractères sont moins supportées, certaines contraintes sont imposées sur leur usage. Elles trouvent leur meilleur usage comme paramètres d'appel des fonctions et procédures.
- Les tableaux d'entiers peuvent avoir jusqu'à trois dimensions. Ils sont toujours alloués dans le dictionnaire, et leur taille n'est limitée que par la quantité de la mémoire vive disponible.
- Les images sont décrites par leur taille et leur profondeur. Elles sont toujours définies et visualisées en trame carrée (c'est le *traitement* qui peut s'effectuer en mode carré ou hexagonal: essayez par exemple la même opération **dil**, sur une même image, en ne faisant varier que la trame).

Il existe aussi un certain nombre de types dérivés, qui se représentent aussi à l'aide d'entiers:

- les éléments structurants,
- les valeurs de pixels,
- les coordonnées de pixels,
- les directions.

Comment coder les éléments structurants ?

Le codage qui suit est destiné à affecter un nombre entier à chaque sous-ensemble de l'hexagone ou du carré unité. Ce code permet d'introduire lesdits sous-ensembles comme éléments structurants dans certaines procédures (*e.g.* **hit-or-miss**). Attribuons au voisinage d' un pixel les poids suivants:

en trame hexagonale	en trame carrée
64 2	256 2 4
32 1 4	128 1 8
16 8	64 32 16

Un élément structurant est codé par la somme des poids de tous les pixels qui le constituent. Par exemple, l'élément structurant:

```
  +   +
*   +   *
  +   +
```

constitué des pixels " + ", est codé comme valant 91 = 64 + 2 + 1 + 16 + 8.

Comment coder les directions ?

Les directions de la grille de travail sont codées de la façon suivante:

en trame hexagonale	en trame carrée
6 1	8 1 2
5 0 2	7 0 3
4 3	6 5 4

Toute direction représentée par une valeur n'appartenant pas à [0..6] en trame hexagonale, ou à [0..8] en trame carrée, est illicite et provoque un message d'erreur.

Quelles profondeurs pour les images ?

Les images sont bi-dimensionnelles et peuvent avoir une des trois profondeurs suivantes:

- 1-bit-per-pixel. Elles sont appelées " images binaires "et prennent les valeurs 0 ou 1;
- 8-bit-per-pixel. Chaque valeur est un entier 8-bit, non-signé, dans l'intervalle [0, +255] ;
- 16-bit-per-pixel. Chaque valeur est un entier 16-bit, signé, dans l'intervalle [-32768, +32767].

On note **min** et **max,** respectivement, les valeurs minimales et maximales de l'intervalle utilisé. D'où le tableau suivant:

	min	max
images binaires	0	1
8-bit images	0	255
16-bit images	-32 768	32 767

Vous pouvez connaître les valeurs de **min** et **max** d'une image donnée au moyen des fonctions **impixmin** et **impixmax.** On notera que, pour des images 16-bit, les valeurs 0 et -32768, sont leurs propres négatifs. La valeur -32768, qui est considérée comme " l'infini " pour les images 16-bit, est positive et négative à la fois.

Les fonctions qui admettent des valeurs de pixels en paramètres d'entrée (*i.e.*: **imwritepix, imdrawline, imset, imcadd, imcsub, imcmul, imcdiv, imcopyngb, immask,** etc..) considèrent toujours ceux-ci comme représentés de façon «générique». La représentation générique d'un pixel est un entier de 32 bit, signé, et prenant les valeurs allant de **min** à **max**.

Dans certaines de ces opérations, si l'on a introduit des valeurs de paramètres invalides, elles sont converties sans que l'erreur ne soit indiquée. Par conséquent, c'est à la routine appelante de fournir correctement les données (*i.e.* en représentation générique). Par exemple, quand on exécute des opérations sur les images binaires, toute valeur, autre que zéro, est considérée dans sa représentation générique comme valant ''1'' et est convertie d'une façon appropriée. C'est pourquoi, en exécutant **imset -1 im1** pour l'image binaire **im1,** on obtient le même résultat qu'avec **imset 1 im1**.

Dans les opérations sur les images 8-bit, les valeurs négatives sont converties en valeurs non signées et tronquées si nécessaire. Ainsi, l'exécution de **imset -1 im1** pour une image 8-bit **im1** donne le même résultat que **imset 255 im1**.

Quand un mot primitif doit renvoyer une valeur de pixel ou la récupérer à usage interne, cette valeur est, elle aussi, représentée par un entier signé 32-bit compris entre **min** et **max.** La conversion interne se produit, par exemple, lors de l'addition d'une image binaire à une autre, à niveau de gris: l'opération équivaut à l'addition de deux images de niveaux de gris, dont l'une n'admet que les valeurs 0 et +1.

Peut-on utiliser des images couleur?

MICROMORPH® ne peut pas traiter directement des images couleur car les opérateurs morphologiques ne sont en général pas définis sur ce type d'images (comme beaucoup d'opérateurs de traitement d'image, la morphologie mathématique travaille sur des scalaires et non sur des données vectorielles telles celles qui

correspondent à des couleurs). C'est la raison pour laquelle vous recevrez un message d'erreur chaque fois que vous tenterez de charger une image BMP 24 bits ou 32 bits (rappelez-vous que les seules profondeurs d'image acceptées par MICROMORPH® sont 1, 8 ou 16 bits). Pour traiter une image couleur avec MICROMORPH®, il faudra extraire préalablement l'information utile de cette image. Ce peut être la luminance ou tout autre scalaire extrait de l'espace de représentation (la teinte, la saturation, la chrominance, etc.). Cependant ces fonctionnalités d'extraction n'existent pas dans MICROMORPH®. En effet, il existe tant de représentations de la couleur que l'incorporation dans MICROMORPH® des routines de conversion et/ou d'extraction des composantes de couleur correspondant à chaque représentation aurait été une tâche énorme et de surcroît inutile car de nombreux logiciels existent pour faire cela. Il est donc de votre responsabilité de préparer les images que vous traitez avec MICROMORPH® de façon à ce qu'elles représentent des scalaires. Cette préparation peut d'ailleurs être faite sans quitter MICROMORPH® en appelant des programmes externes (voir comment dans cette FAQ) et en transférant les images par le presse-papier (opération également expliquée dans cette FAQ).

Inversement, on pourrait penser que le traitement par MICROMORPH® d'images couleur 8 bits (256 couleurs) ne présente aucune difficulté. C'est en effet vrai en ce sens que le chargement dans MICROMORPH® d'une telle image ne générera aucune erreur. Cependant, il est fort probable que les résultats des transformations n'auront aucun sens! En effet, dans ce cas, la valeur numérique associée à chaque pixel de l'image correspond à un index dans une table de correspondance appelée ***palette de couleur***. Cette palette de couleur définit l'ensemble des couleurs utilisées dans l'image. Elle est contenue dans le fichier-image BMP. Cependant MICROMORPH® ne tient absolument pas compte de cette palette de couleur. Seule la valeur scalaire du pixel est utilisée. Une même valeur pourra correspondre à des couleurs totalement différentes selon la palette utilisée. C'est pourquoi la seule palette qui a réellement un sens dans MICROMORPH® au niveau du traitement est la palette de gris. Dans ce cas, un pixel de valeur n aura comme couleur le vecteur RGB égal à (n,n,n) c'est-à-dire en fait la valeur de gris n. Si vous visualisez une image couleur 8-bit dans MICROMORPH®, cette image sera d'ailleurs visualisée comme une image à niveaux de gris. L'aspect de l'image vous donnera une idée assez exacte des valeurs réelles de chaque pixel utilisées par MICROMORPH®. Faites le test et vous ne manquerez pas d'être, dans la plupart des cas, surpris par le résultat!

Comment utiliser des palettes de couleur?

Vous pouvez améliorer la visualisation des images dans MICROMORPH® en leur associant des palettes de couleur. Au moment du lancement initial, MICROMORPH® cherche un fichier contenant la palette de couleurs par défaut. Ce fichier doit s'appeler DEFAULT.PAL, et il doit se trouver dans le répertoire défini par la variable d'environnement PAL dans le fichier de configuration MMORPH.INI. Si ce fichier n'est pas trouvé ou s'il y a une erreur de format, le logiciel calcule une palette interne contenant 256 niveaux de gris, et cette dernière sera utilisée comme palette par défaut.

La palette par défaut affecte l'apparence initiale de toutes les images affichées: elle est dès le début attachée à toutes les fenêtres affichant les images et a un " handle "(ou étiquette) égal à 1.

Notez ici qu'il ne s'agit pas d'images (qui ne " savent " pas que les palettes existent, voir le traitement d'images couleur dans cette FAQ), mais de leurs fenêtres d'affichage. Ce sont ces dernières auxquelles la palette n°1 s'attache au moment de **imdisplay** et qui peuvent en recevoir une autre, à l'aide de **imattach**. Une palette s'introduit quand on veut afficher l'image. Par contre, quand on colle une image à partir du presse-papiers ou quand on la charge de disque, la palette stockée dans l'image BMP source n'est jamais consultée. Elle ne peut nullement affecter l'affichage de son image ou d'une autre (voir encore une fois le traitement des images couleur).

De la même façon, quand une image est sauvegardée sur le disque ou copiée dans le presse-papiers, l'image BMP résultante possède toujours la palette "niveaux de gris" et non pas celle attachée à la fenêtre d'affichage, s'il y en a une. La palette attachée n'est qu'une caractéristique de visualisation de l'image à l'intérieur de MICROMORPH® et n'est jamais exportée.

Dès que l'image est affichée dans sa fenêtre, on peut associer une autre palette à cette fenêtre. Afin de le faire, il faut avoir la palette nécessaire déjà chargée du disque par intermédiaire de **imloadpal**. Cette fonction renvoie le " handle " à utiliser dans l'appel à **imattach**.

Une fois chargée, la palette est conservée dans la mémoire de MICROMORPH® jusqu'à ce qu'elle soit explicitement détruite, et pendant ce temps-là elle peut être attachée à plusieurs fenêtres. On peut effacer une

palette de la mémoire MICROMORPH® à n'importe quel moment en appelant **imdiscard**, quelque soit le nombre d'attachements qu'elle a reçu. Cependant, on ne peut pas détruire la palette par défaut (numéro 1).

Pour détacher la palette d'une fenêtre sans l'effacer, il suffit de la remplacer par celle par défaut:

imattach 1 imin

C'est exactement ce qui se passe quand on détruit la palette qui a été attachée à une ou à plusieurs fenêtres.

En modes vidéo 8 bits (ceux qui affichent au maximum 256 couleurs en même temps), à chaque moment on ne peut avoir qu'une seule fenêtre affichant les couleurs correctes, à savoir celle qui est " active ". Par conséquent, pour obtenir l'affichage correct, il faut " activer " la fenêtre nécessaire, en cliquant sur elle ou sur son cadre.

Ceci n'a aucune importance, une fois dans le mode video 16- ou 24-bit et plus.

Lorsqu'une image en couleurs est définie par trois images numériques (ex RGB, HLS, etc..), ses traitements sont réalisables à l'aide des opérations numériques de MICROMORPH®. Sa visualisation en revanche n'est pas directement possible, car elle nécessite de réduire l'image à 256 couleurs. Cela peut être obtenu par un logiciel externe qui produit à la fois l'image aux couleurs réduites et la palette qui leur est affectée. Ces palettes ont généralement pour format la succession des 256 triplets de couleurs, précédée d'un en-tête. Il faut supprimer cet en-tête, et sauvegarder la palette brute dans un fichier "pal" de MICROMORPH®. Dès lors, la visualisation de l'image s'obtient en faisant:

col in "palette" { affecte à la mémoire image "in" la palette sélectionnée}
imload in "image réduite" { charge dans "in" l'image réduite à 256 niveaux}

Par ailleurs, un certain nombre de palettes standard sont livrées sous forme de fichiers "pal" dans MicroMorph, et permettent un certain nombre de visualisations en couleur, telles que "dscol" (superposition colorée de deux images), "pscol" (labellisation et colorisation des différentes composantes connexes d'une image binaire), "profile" (tracé du profil de gris d'une image numérique) , etc..

Selon quels modes MICROMORPH® fonctionne-t-il ?

Un mode est un choix de traitement, qui perdure à travers les instructions jusqu'à ce qu'on indique explicitement d'en changer. MICROMORPH® dispose essentiellement de trois modes, qui sont indépendants entre eux.

Il s'agit:

1 - du choix de la trame, carrée ou hexagonale;
2 - du traitement des bords de champ, de manière euclidienne ou standard;
3 - de la taille des images traitées.

Les deux premiers modes portent sur un choix binaire, où l'option par défaut, au lancement d'une session de travail, est la trame hexagonale avec traitement standard des effets de bords.

La taille des images traitées s'exprime d'une part par le format d'image, et, en plus, par le choix d'une fenêtre de travail. Par défaut, ces deux paramètres sont égaux et valent 256*256 pixels.

Les choix modaux sont exclusifs: on ne peut pas avoir au même moment des traitements en trames carrée et hexagonale, par exemple. En ce sens , la profondeur de bits des images ne constitue pas un mode, et l'on peut très bien travailler simultanément avec des images binaires et numériques.

Comment indiquer et/ou modifier une taille d' images ?

Les tailles horizontale et verticale des images sont deux modes contrôlés par le mot primitif **iminit**, ou aussi bien, par le mot composé **format**. Cette dernière commande alloue également un certain nombre d'images numériques et binaires par défaut. La taille minimale permise en dimension X est 128 pixels. Elle doit toujours être un multiple de 32. La taille minimale en dimension Y est 2 et doit toujours être paire. En principe, les dimensions d'image peuvent s'étendre à 16384 x 16384. En pratique, la mémoire vive disponible risque de

restreindre ces dimensions maximales. Lorsqu'on appelle une image plus grande que le format en cours, c'est la partie supérieure gauche de l' image qui est chargée.

La zone de travail à l'intérieur de l'image est aussi un mode, contrôlé par le mot primitif **imsetwindow**. Par défaut, les dimensions de la fenêtre de travail coïncident avec celles de l'image.

Comment lire les coordonnées des pixels ?

Le pixel du coin haut gauche a les coordonnées (0,0). La coordonnée X s'incrémente de gauche à droite, et Y de haut en bas. On suppose donc que les images sont entièrement placées dans le premier cadran du système d'axes. Par conséquent, toute coordonnée négative pointant vers un pixel non existant sera rejetée, ou convertie, selon l'opération impliquée (voir **imwritepix, imreadpix**).

Trame carrée ou hexagonale ?

Les images sont traitées en trame carrée ou en trame hexagonale. Le choix de la trame est contrôlé par le mot primitif **imsetgrid**:

- W **imsetgrid 0** établit la trame carrée pour toutes les images;
- W **imsetgrid 1** établit la trame hexagonale pour toutes les images.

Le type de la trame s'applique à l'exécution de seulement six opérations de voisinage, à savoir: **imcopyngb, iminfngb, imsupngb, imdiffngb, imbuildngb** et **imhitormiss**. Le type de la trame est associé plutôt à ces opérations qu'aux images mêmes. Ainsi, toutes les images allouées après un appel à **imsetgrid** auront le type de la trame défini par cet appel, et le prochain appel à **imsetgrid** changera le type de la trame pour les images déjà allouées.

Pour les deux types de trame, la coordonnée Y croît du haut en bas; la coordonnée X croît de gauche à droite; le pixel du coin en haut à gauche ayant les coordonnées (0,0). Dans le cas de la trame hexagonale, chaque ligne de l'image a sa parité associée: on dit que les lignes de coordonnée Y paire, à commencer par 0, sont dites " paires ", les autres " impaires ". Ceci n'a rien à voir avec la longueur des lignes, mais correspond au schéma suivant:

```
ligne 0 :  * * * * * * *
ligne 1 :    * * * * * *
ligne 2 :  * * * * * * *
                etc...
```

On peut se renseigner sur la trame en cours par l'intermédiaire du mot primitif **grid**.

Comment gérer les bords de champ ?

Quel que soit leur type, les images possèdent toujours un bord externe, formé d'une ligne au-dessus et une ligne au-dessous de l'image, d'une colonne à gauche et une colonne à droite. Ce bord est invisible, il ne s'affiche pas à l'écran. Il n'affecte que le fonctionnement des cinq opérations liées au voisinage: **iminfngb, imsupngb, imdiffngb, imbuildngb, imhitormiss**. Chacune d'elles, au moment de son exécution, préétablit les valeurs du bord de façon appropriée (il n'existe toutefois aucune image mémorisée des bords ou de l'extérieur du champ).

opération	valeur de **edge**	
	0	1
iminfngb	**min**	**max**

imsupngb	**min**	**min**
imdiffngb	**min**	**min**
imbuildngb	**min**	**min**
imhitormiss	**min**	non attribué.

La valeur attribuée à l'extérieur est contrôlée par le mot primitif **imsetedge**, selon le tableau ci-dessus où **min** et **max** désignent les valeurs extrémales du type d'image en cours, *i.e.* 0 et 1 pour les images binaires, 0 et 255 pour les images 8-bit, et -32768 et + 32767 pour les images 16-bit .

L'effet de **imsetedge** porte sur toute l'image ou, aussi bien, sur la zone d'intérêt définie par l'intermédiaire de **imsetwindow**. Si cette zone est strictement plus petite que le format des images, les pixels qui lui sont extérieurs ne sont jamais modifiés. A tout moment, on peut se renseigner sur la valeur courante de l'effet de bord en utilisant la fonction **edge.** On notera que **imsetedge** and **imsetgrid** sont deux opérations indépendantes.

Comment exécuter une procédure ou une fonction ?

Les mots de MICROMORPH® se classent en deux catégories. Ceux qui renvoient un nombre sont appelés "fonctions" et les autres "procédures". Les unes et les autres peuvent s'exécuter de deux manières :

1 - On peut les introduire au clavier dans la fenêtre de commandes, puis taper la touche **enter** ;

2 - On peut créer une procédure ou une fonction nouvelle, ce qui implique de lui donner un nom. La procédure est rédigée sous forme d'un fichier texte à l'aide d'un éditeur de texte quelconque, puis compilée. Son nom est alors intégré comme mot supplémentaire dans le Dictionnaire. Si le même mot y était déjà utilisé, la nouvelle définition remplace toujours automatiquement l'ancienne.

Comment créer une procédure ou une fonction?

Un des avantages de MICROMORPH® est de vous permettre d'ajouter au dictionnaire vos propres procédures, après les avoir définies en utilisant des procédures et des fonctions déjà disponibles. Cette structure de programmation est dite chaînée. Pour définir de nouveaux mots, le plus simple consiste à écrire leur définition dans un fichier texte en utilisant l'éditeur de texte de votre choix (comme le bloc-notes de Windows®, le traitement de texte WordPad ou tout autre traitement de texte capable d'enregistrer un fichier sous forme de texte simple).

On utilise **deproc** pour créer une procédure nouvelle, et **defunc** une fonction nouvelle**.** Les mêmes commentaires s' appliquant à ces deux procédures, nous ne traitons en détail que la première.

Syntaxe
deproc procname **pre-params** procname **post-params**
syntax "commentaires utiles**"**
  sequence d'instructions
end

Effet
Définit une nouvelle procédure dont le nom est *procname*, équipée avec pré-paramètres ou post-paramètres ou les deux.

Il est nécessaire d'indiquer deux fois le nom du mot qu'on est en train de créer: la première fois, tout de suite après **deproc,** pour le déclarer, et la deuxième fois, en l'utilisant comme séparateur, entre la liste des pre-paramètres et celle des post-paramètres. Cette présentation est explicitée par l'opérateur **syntax**; c'est pourquoi ce dernier est absolument indispensable juste après définition du mot.

Le mot **syntax** permet aussi d'introduire des commentaires, qui sont alors insérés dans le Dictionnaire. Si l'on n'y tient pas , on peut se contenter d'une version minimaliste en tapant **syntax '' ''**.

Comment introduire des paramètres dans une procédure ?

On peut attribuer des valeurs aux paramètres d'une fonction (ou d'une procédure) " par valeur " ou " par référence ". Illustrons ce point à l'aide d'un exemple simple en considérant une procédure P qui calcule le double d'une valeur numérique d'entrée et l'imprime à l'écran:

```
deproc P P v
syntax "P valeur -> imprime le double de la valeur "
v := ( 2 * v )
print v
end
```

Appliquons cette procédure à une variable A qu'on a déclarée par **int A ;** et à laquelle on a attribué la valeur 5: **A := 5.** Dans ce cas, **P A** imprime "10" à l'écran. **P 5** donne le même résultat. Si, maintenant, nous imprimons A avec **print A**, on constate que sa valeur reste inchangée et toujours égale à 5. On dit que la variable *A* a été passée ***par valeur*** , puisque sa valeur est simplement copiée dans *v* lorsqu'on fait appel à la procédure.

Redéfinissons maintenant la même procédure, mais en notant le paramètre *v pour spécifier qu'il doit être passé ***par référence:***

```
deproc P P *v
syntax "P  valeur -> imprime le double de la valeur"
*v := ( 2 * *v )
print *v
end
```

Comme avant, **P A** imprime ''10'' à l'écran. Mais cette fois-ci, si l'on fait **print A** on voit que la valeur de *A* est devenue 10. Outre cela, l'exécution de **P 5** provoque maintenant un message d'erreur disant qu'une variable est attendue dans P. Ceci signifie que la valeur de *A* n'est plus copiée dans *v* : son adresse est passée à P, où elle est utilisée en tant qu'adresse de *v*.

Exemple

```
deproc env env
syntax "env (no parameters) -> prints some vital values"
print [ "edge = " edge ]
print [ "grid = " grid ]
print [ "connectivity = " connectivity ]
end
```

Comment compiler un fichier ?

Lorsque vos nouvelles procédures ou fonctions ont été créées (vous pouvez en mettre plusieurs dans un même fichier), il vous suffit alors de compiler le fichier à l'aide de la commande ***file/Compile*** de la barre de menu de la fenêtre de l'interpréteur et, si vous n'avez pas fait d'erreurs, vous verrez vos procédures s'ajouter au dictionnaire. Ces nouvelles procédures peuvent alors être utilisées à votre guise, soit en les lançant à partir de l'interpréteur, soit en les utilisant dans la définition de procédures encore plus complexes.

Alternativement, on peut aussi compiler le fichier à l'aide de la procédure **comp**:

Syntaxe

comp "nomfich**"**

Effet

Compile le fichier *nomfich*. La compilation d'un fichier consiste en l'interprétation de son contenu comme s'il était tapé ligne par ligne dans la fenêtre de commandes. Les fichiers scripts doivent avoir une taille plus petite que 65536 octets. Les lignes plus longues que 255 caractères seront tronquées sans indication d'erreur.

Comment ajouter des procédures au démarrage de MICROMORPH ?

Pour vous éviter de compiler vos fichiers à chaque démarrage de MICROMORPH®, il vous suffit d'insérer une commande de compilation de ces fichiers dans le fichier ***com.mic***. Ce fichier contient en effet l'ensemble des fichiers compilés par MICROMORPH® au démarrage. En fait, MICROMORPH® exécute le fichier ***init.mic*** qui compile le fichier ***com.mic*** qui lui-même compile les autres fichiers. Si votre fichier de procédures personnelles se nomme **myprog.mic**, il suffit d'ajouter à la fin de ***com.mic*** la ligne :

comp "myprog"

et les procédures et fonctions définies dans **myprog.mic** seront automatiquement compilées chaque fois que vous lancerez MICROMORPH®. Vous pouvez également en éditant le fichier ***com.mic*** modifier le dictionnaire initial de MICROMORPH®. Cependant, n'entreprenez ces modifications que si vous êtes sûr de respecter le chaînage des opérateurs. Rappelez-vous en effet que la suppression d'un fichier peut entraîner des erreurs de compilation, car les fichiers suivants risquent de ne plus trouver certaines procédures qui leur sont indispensables.

Comment insérer une aide en ligne dans une procédure ou une fonction?

Pour les utilisateurs avancés!

On peut provoquer l'exécution d'une programme externe à l'aide du mot **syntax.** Pour cela, il faut d'abord définir le programme extérieur. On le fait par une chaîne entre guillemets qui commence par un point d'exclamation, et continue par le nom du programme, suivi d'éventuels paramètres. Puis on compile la procédure. Pour lancer le programme ainsi défini, il suffit maintenant de cliquer sur la procédure en question dans la fenêtre du dictionnaire. Par exemple, lorsqu'on rédige un programme dont le code source est placé dans **fichier.mic**, on peut y accéder à tout moment en faisant

syntax '' ! notepad fichier.mic ''

Autre exemple: pour que l'on puisse appeler l'aide en ligne Windows de cette façon, un petit exécutable est fourni avec MICROMORPH®, **whelpdrv.exe:**

syntax '' ! whelpdrv mmorph.hlp imabsmax ''

Vous pouvez remplacer le fichier d'aide par n'importe quel fichier d'aide que vous aurez créé vous-même pour commenter la procédure ou la fonction que vous venez de définir. Vous pouvez aussi, si la réalisation d'un fichier d'aide ne vous semble pas une tâche facile, utiliser d'autres moyens. Vous pouvez par exemple décrire et commenter votre procédure dans un fichier HTML (appelé par exemple **monprog.htm**) et le visualiser lorsque vous cliquez sur le nom de la procédure dans le dictionnaire par le biais de l'instruction:

syntax '' ! iexplore monprog.htm''

si Internet Explorer est votre navigateur web et si son chemin d'accès a été spécifié au démarrage.

Comment appeller un programme externe à partir de MICROMORPH®?

Vous pouvez lancer l'éxécution d'une application Windows à partir de MICROMORPH® à l'aide de la commande **!** suivie du nom du programme appelé. Cette commande lance l'exécution de l'application Windows et retourne tout de suite à l'interpréteur. Elle ne retourne jamais aucune valeur. L'interpréteur n'a aucun moyen de détecter la terminaison d'une application lancée de cette façon. Cela signifie que les instructions MICROMORPH® suivant l'instruction **!** seront éxécutées immédiatement. Si vous attendez un quelconque retour (image, valeur...) du programme externe, il faudra veiller vous-même à insérer une pause dans le programme MICROMORPH® pour

être sûr qu'il ne reprenne que lorsque le programme externe sera terminé. Avec "**!**" on ne peut lancer que les applications existantes sous la forme d'un fichier exécutable sur le disque (on ne peut pas, par exemple, exécuter de commandes implantées dans command.com). Si le fichier exécutable ne se trouve sur le PATH, on doit fournir son chemin d'accès complet, par exemple:

! c:\user\mapsym.bat

Comment utiliser le presse-papier?

Le presse-papier de Windows peut être utilisé avec MICROMORPH® pour récupérer des images venant de programmes externes ou au contraire pour envoyer des images MICROMORPH® vers des programmes externes. Cette facilité est particulièrement intéressante pour récupérer par exemple des images en provenance d'un logiciel de capture ou de retouche graphique ou encore pour envoyer des images vers des programmes de traitement externes disposant de procédures absentes dans MICROMORPH® (par exemple des fonctions de séparation ou de concaténation des diverses composantes d'une image couleur). On peut utiliser le presse-papier soit par l'intermédiaire du menu ***edit*** dans la barre de menu, soit à l'aide des commandes **imclipcopy** et **imclippaste** qui peuvent être utilisées dans l'interpréteur de commandes ou dans la définition d'une procédure ou d'une fonction.
Attention! Vous ne pouvez transférer que des images au format Windows DIB (format BMP 8 bits) sinon vous obtiendrez un message d'erreur. D'autre part, la palette de couleur de l'image est perdue quelque soit le sens du transfert (vers MICROMORPH® ou en sortant du logiciel). La seule palette conservée est la palette à ***teintes de gris*** (cf. les rubriques sur les images couleur et les palettes).

Quelles boucles logiques dans une procédure?

MICROMORPH® dispose des boucles logiques **for**, **if** et **while**, dont les syntaxes sont les suivantes:

for

Syntaxe
for expression1 **to** expression2 **do** sequence-d'instructions **end**

Effet
Exécute *sequence-d'instructions* dans la boucle (*expression2* - *expression1* + *1*) fois. Les deux expressions sont évaluées uniquement une fois, avant d'entrer dans la boucle. N'importe quelle boucle peut être interrompue avec Control-C. Les mots **do** et **end** sont indispensables, même quand *sequence-d'instructions* est composée d'un seul mot. De plus, le mot **do** doit se trouver sur la même ligne que **for**.

Exemple

```
for 1 to 0 do print "nothing" end
int a N ;
a := 1 N := 20
for a to N do
   print "N times"
end
```

if

Syntaxe
if expression **then** sequence-d'opérateurs **end**
if expression **then** sequence-d'opérateurs_1 **else** sequence-d'opérateurs_2 **end**
if expression **then else** sequence-d'opérateurs **end**

Effet
Opérateur conditionnel.

Exemple

```
if (a = 1) then print "a equals 1"
else print "a does not equal 1"
end
```

while

Syntaxe
while *expression* **do** *sequence-d'opérateurs* **end**

Effet
Exécute *sequence-d'opérateurs* tant que la valeur de *expression* n'est pas 0. Toute boucle, même infinie, peut être interrompue par Control-C. Les parties **do** et **end** sont indispensables et doivent être toujours présentes dans la construction, même si la *sequence-d'opérateurs* est composée d'un seul opérateur. De plus, le mot **do** doit se trouver sur la même ligne que **while**.

Exemples

1/
```
while 1 do print "infini" end { imprime " infini " indéfiniment }
```

2/
```
int a N ;
a := 0
N := 20
while (a < N ) do
print cat ["a = " a ]
a := ( a + 1 )
end                { la procédure  imprime 20 fois ''a = '' suivi de la valeur de a }
```

3/
```
int w2 ; w2 := imalloc 8 imcopy g1 w2
 while ( imvolume w2 ) do
{ séquence d'opérations }
end          { la séquence d'opérations est itérée  jusqu'à ce que le volume (l'aire) de w2 soit égal à 0. }
```

Comment allouer des espaces images dans une procédure 2D?

On a très souvent besoin de créer des espaces mémoires pour stocker des images de travail. La procédure à utiliser se nomme "imalloc"

Syntaxe
imalloc pixdepth

Effet
Alloue une image de la profondeur spécifiée et renvoie son ''handle''. La profondeur doit être 1, 8 ou 16. L'image est créée avec les dimensions établies par le dernier appel à **iminit.** A tout moment, on peut savoir la profondeur d'une image existante en appelant **imdepth.** Les images allouées à l'intérieur d'une procédure définie par **deproc** ou **defunc,** sont automatiquement détruites à la sortie de la procédure. La seule exception à cette règle se produit quand a lieu un appel à **iminit** dans la procédure en question. Dans ce cas, les images allouées après cet appel sont conservées (cette caractèristique est utilisée dans la procédure **format**).

Exemple
```
deproc top top s d sz                          { s et d n' ont pas à être introduits comme espaces mémoire }
syntax "grad source destination taille : top-hat par ouverture"
   int u v ;
   u := imalloc 8     v := imalloc 1
   open s u  sz
   imdiff s u d
```

```
  imthresh d 0 5 v
  imdisplay v " tophat seuillé"
end                                    { les espaces u et v sont désalloués, l' image v disparait de l' écran}
```

Comment allouer des espaces images dans une procédure 3D?

Les procédures 3D des fichiers *3dutil.mic* et *3dproces.mic* demandent qu'on alloue préalablement des zones mémoire pour les images des séquences à étudier. Or le fichier *init.mic* impose déjà une première série d'allocations de 10 mémoires images de 8 bits, suivies de 10 autres binaires et enfin de 3 images numériques de 16 bits. Si l'on fait donc:

```
int i ;
i := imalloc 1
```

on alloue un espace mémoire d'un bit de la taille d'une image, mémoire dont l'étiquette est 24 (il suffit de faire:
ds 24
pour visualiser cette mémoire).
Plus généralement, si l'on a besoin de 60 mémoires de 8 bits par exemple, pour une séquence de 30 images et pour 30 images transformées, alors, avant de faire appel à **Seqload**, on écrira:

```
int i ;
for 1 to 60 do
i := imalloc 8
end
```

ce qui crée 60 allocations numérotées de 24 à 83.
Si maintenent on lance:

```
Seqload "c:\études\seq\" 8 30 10
```

les opérations suivantes sont effectuées

```
imload 30 "c:\études\seq\8.bmp"
imload 31 "c:\études\seq\9.bmp"
....
imload 39 "c:\études\seq\17.bmp"
```

Si l'on est amené à écrire soi-même des opérations tridimensionnelles, on a tout intéret à créer les zones mémoires à l'intérieur des procédures que l'on construit, de façon à ce qu'elles s'effacent automatiquement en fin de procédure et ne grèvent pas la mémoire vive de l'ordinateur.

Comment afficher un résultat ?

L'instruction de base pour afficher un résultat est print. Lorsqu'elle est associée à l'instruction output, elle permet de définir où le résultat est affiché.

Syntaxe

print expression
print **[** expression1 ... expressionN **]**

Effet

Cette procédure transfère à sa destination une (la première forme) ou plusieurs valeurs d'expressions concaténées (la seconde forme). La destination peut être l'écran ou le fichier sur disque (voir le mot **output**).

Exemple
print "character string" { affiche à l'ecran ‘’character string ’’ }
int i; print i
print ["i = " i] { affiche à l'ecran : ‘’i := ‘’ suivi de la valeur de i }

Comment stocker des résultats, et éditer des graphiques?

En règle générale pour les fonctions de MICROMORPH®, les valeurs ne sont que stockées dans une variable, puis affichées par le biais de **print**. Si l'on veut de plus les conserver, il faut faire précéder la fonction de l'ouverture d'un fichier et la faire suivre de la fermeture de celui-ci. Ainsi, si l'on veut connaître l'évolution du périmètre des érodés de l'image b1, on écrira:

```
output "c:\b1.dat"
while  (area b1) do
print perim b1
ero b1 b1 1
end
output " "
```

La dernière instruction redéfinit l'écran comme destination par défaut.

Toutefois, la plupart des fonctions de la section 15 du manuel de référence (courbes) offrent un raccourci, car elles admettent un nom de fichier en variable d'entrée. Par exemple, pour la granulométrie des images lung1 et lung2, de 1 en 1 jusqu'au pas 25, on écrira, après avoir chargé les images en g1 et en g2 :

```
granul  g1  1  25 "lung1.dat"          granul  g2  1  25  "lung2.dat"
```

Les valeurs granulométriques sont alors enregistrées dans les fichiers *lung1.dat* et *lung2.dat*. Elles peuvent servir pour tout traitement ultérieur. En particulier, si l'on veut les représenter graphiquement, on peut utiliser un programme externe comme par exemple ***gnuplot*** où la suite d'instructions suivantes:

```
set xlabel "SIZE"  0        set ylabel "AMOUNT" 0
 set title "Grey size distribution (by openings)"
plot "c:\wmmorph\lung1.dat" with linespoint , "c:\wmmorph\lung2.dat" with linespoint
```

produit le tracé des deux courbes granulométriques sur un même graphique.

Qu' est-ce-que le Dictionnaire ?

Le dictionnaire contient la liste de tous les mots (fonctions et procédures) disponibles. Ces procédures et fonctions sont compilées au démarrage. Elles sont présentes dans le noyau du logiciel ou contenues dans les fichiers avec l'extension **.mic**. L'initialisation de MICROMORPH® vous fournit un nombre considérable de procédures. Il existe en fait deux types de mots. Les mots primitifs constituent le noyau de base de MICROMORPH®. Ils ne sont pas modifiables. S' ajoute à ce noyau de base la collection de mots composés déjà préparés pour l'utilisateur. Ces mots composés ont été élaborés à partir de mots primitifs, ou de mots composés plus simples, comme il est indiqué en *«Comment créer une procédure?»* . Vous pouvez à votre tour enrichir le dictionnaire, et le personnaliser de vos propres mots de façon très simple. Il suffit de compiler les fichiers-source correspondants. La structure de programmation du langage MICROMORPH® est relativement simple et l'ajout de nouvelles transformations se fait sans difficulté.

On ouvre le dictionnaire en cliquant sur l' icône correspondante, qui est visualisée en permanence. En cliquant ensuite sur un mot, on obtient sa syntaxe, ou s'il s'agit d'un mot primitif ou d'un mot auquel on a associé un fichier d'aide (voir la rubrique correspondante), une description de son effet, ses caractéristiques de fonctionnement (modes, profondeurs d'image, ..), éventuellement un exemple et une liste de mots voisins. Les caractéristiques de fonctionnement sont précisées par les conventions suivantes :

grid : 1 trame hexagonale exclusivement;
grid : 0 trame carrée exclusivement;
grid : 1,0 trame carrée ou hexagonale;
grid : [1],0 basculement automatique en carré pour la procédure, et retour à la grille antérieure (*mutatis mutandis* pour *grid* : 0, [1]).
edge : 1 mode standard exclusivement;
: 0 mode euclidien exclusivement, intrinsèque (op. croissantes) ou transitif (amincissement);
edge : [1],0 basculement automatique sur **imsetedge** = 0 le temps de la procédure, et retour à l'état antérieur (*mutatis mutandis* pour *edge* : [0], 1).
depth : bin ; grey ; bin to grey, etc... ce terme n'est pas associé à un mode, comme *grid* et *edge*, mais indique la profondeur de bits des images traitées.

Par exemple, la *même* procédure **dil** effectue, selon les valeurs des clés, une dilatation hexagonale ou carrée, en mode standard ou euclidien. Et elle est automatiquement binaire ou numérique selon que l'image à traiter est elle- même binaire ou numérique. Enfin, le dictionnaire classant les mots dans l'ordre alphabétique parce que celui-ci est universel (mais pas le plus approprié pour la recherche), nous donnons ci-après une entrée thématique pour les deux séries de mots primitifs et composés.

DICTIONNAIRE THEMATIQUE DES MOTS DE MICROMORPH®

I - Table des procédures et des fonctions primitives

1- MANIPULATIONS D' IMAGES

imalloc, imfree	alloue, désalloue une image de 1-, 8- ou 16-bits
imclippaste, imclipcopy	charge, prend une image bitmap du presse-papier Windows
imcopy, imcopyhibyte, imcopylobyte, imcopyplane	copie, insère une image dans une autre
imdepth, impixmin impixmax, imparam	renvoie les paramètres de taille et de profondeur de l'image
imdrawline	trace un segment dans une image
iminit, imsetwindow	introduit les dimensions de l'image et de la zone de travail
imload, imsave	charge, mémorise une image sous fomat BMP
imwritepix, imreadpix	écrit, lit un pixel dans une image

2 - OPERATIONS PONCTUELLES (SUR TOUTE L' IMAGE)

imabs	convertit les pixels d'une image 16-bit en leur valeur absolue
imabsmax, imabsmin	donne la valeur maximum/minimum des pixels de l'image
imadd	additionne deux images
imcadd	ajoute une constante à une image
imcdiv	divise une image par une constante
imcmul	multiplie une image par une constante
imcompare	compare deux images
imcsub	soustrait une constante d'une image
iminv	inverse une image
immask	interprète une image binaire comme image numérique
imor, imand, imdiff, imxor	opérations booléennes sur des images binaires
imset	crée une image de valeur constante
imsub	soustrait une image d'une autre
imsup, iminf	maximum, minimum de deux images
imthresh	seuillage de l'image
imvolume	somme des valeurs des pixels de l' image

3 - OPÉRATIONS DE VOISINAGE (SUR TOUTE L' IMAGE)

imbuildngb	reconstruction élémentaire
imconvolve	convolution d'une image selon un noyau spécifié
imcopyngb	translation unitaire d'image selon les directions de la trame
imdiffngb	différence ensembliste avec l'image translatée unitaire

imhitormiss	transfomation par tout ou rien
imsupngb, iminfngb	max, min entre l'image et sa translatée unitaire

4 - GESTION DES PROCEDURES ET DE LA VISUALISATION

!	exécute un fichier exécutable
?terminal, readkey	opérations de lecture sur le clavier
comp	compile un fichier d'instructions
del	efface un mot du dictionnaire de MicroMorph
edge, imsetedge	indique, fixe le mode de traiement des effets de bords
grid, imsetgrid	indique, fixe la trame carrée ou hexagonale
imdisplay, imunlock, imshut	
imlock, imscale	outils de visualisation
imloadpal, imattach, imdiscard	maniement de la palette pour la visualisation en couleurs
imtrace, imtracend, imgetcursor	lecture de la position de la souris dans l'image
input, output, read, print, eof	entrée/sortie en indiquant la destination vers fichier ou ecran
makeerror	provoque un message d'erreur
randomize, random	engendre un nombre aléatoire
serotate, setranspose	maniement d'élément structurant
sound	produit un son (plutot laid)

II - Table des procédures et fonctions composées

1 - OUTILS UTILITAIRES

clr	mise à zéro d'une image
power	(arith) fonction puissance
mod	(arith) modulo
div	(arith) division arrondie à l'entier le plus proche
abs	(arith) valeur absolue
inside, inferior	A inclus dans B, $f \leq g$
translate	translation
div2	division par deux d'une image
mean	moyenne de deux images
incrust	incruste une portion d'une image dans une autre
bintogrey	version numérique d'une image binaire
getcoords	conversion de coordonnées
masksup(equal)	masque du domaine $g_1 > g_2$, $(g1 \geq g2)$
greymask	constante dans un masque donné
s4rotate	rotation de 90°
greyabs	valeur absolue d'une image
format	fixe le format des images MICROMORPH
adjust	seuillage interactif
ngbnb	nombre de voisins d'un point sur la trame
dirtranspose	calcule une direction transposée

clean	mise à zéro des basses valeurs
immean	valeur de gris moyenne d'une image
getcoords	retourne les coordonnées d'un point
gandb	intersection entre une image binaire et numérique
stop	arrêt et attente de pression d'une touche du clavier
stretch	étend l'intervalle de gris d'une image
delta	masque des différences entre deux images
half	réduit d'un facteur 2 la taille d'une image

2 - OUTILS DE VISUALISATION

ds dscol	abréviations pour la visualisation
shadow shadow2	ombrage
clrcol	désactive une palette de couleurs
col	affecte une palette de couleurs
pscol	étiquetage de couleurs
cp	copie d'une image avec sa palette
cppal	copie une palette
refresh	visualise une image au-dessus des autres
profile	visualisation d'un profil de gris
coldisplay	visualisation couleur

3 - EROSIONS, DILATATIONS

3.1. Erosions et dilatations de base

dil, ero	dilatation, érosion, carrée ou hexagonale
dirdil, direro	dilatation, érosion par un segment
minidil, miniero	dilatation, érosion, par petit carré, triangle
distance	fonction distance carrée ou hexagonale
dbldil, dblero	dilatation, érosion par doublet de points
cont, sq4cont	contours internes de l'image
gradient, sobel	gradients morpho et de Sobel

3.2. Autres dilatations et érosions

isodil, isoero	dilatation, érosion, isotrope
isodist	fonction distance isotrope
conedil, cyldil	dilatation conique, cylindrique
crossdil	dilatation d'un ensemble par un autre
rhombodil, rhomboero	dil., érosion, par un rhombododécaèdre
diamdil, diamero	dilatation, érosion par un diamant
ringdil1, ringdil2	dilation par un anneau hexagonal ou par ses sommets
(routines intermédiaires)	dilatations polygonales

4 - OPERATEURS DE RANG

hexrank	opérateur de rang hexagonal
median	médiane (carré, hexagonal)
binsegmi	intersection de dil. par des segments

5 - CONVOLUTIONS

vgauss1, hgauss1, gauss1	filtres gaussiens vert., hor. et isotrope de taille 1
gauss	filtre gaussien de taille n

6 - OUVERTURES, FERMETURES

6.1. ouvertures, fermetures de base

open, close	ouverture, fermeture sur dil et ero
diropen, dirclose	ouverture, fermeture sur dirdil et direro
lineopen, lineclose	réunion, (intersection) des précédentes
openth, closeth	top hats sur open, close
lineopenth, linecloseth	top hats sur lineopen, sur lineclose

6.2 Autres ouvertures et fermetures

isopen, isoclose	ouverture, fermeture, isotrope
infopen, supclose	limite d'inf d'ouvertures directionnelles (ou de sup de fermetures)
miniopen, miniclose	ouverture, fermeture sur minidil, miniero
stopen	inf d'ouvertures directionnelles

7 - GEODESIE ET CONNEXITE

gdsdil, gdsero	dilatation, érosion, géodésique
build	reconstruction binaire par marqueur
buildopen, buildclose	ouverture, fermeture, par reconstruction
areaopen	ouverture selon l'aire
ringbuild	reconstruction par un anneau
gdsdist	distance géodésique
recons	reconstruction binaire (similaire à build)
levelling	nivellement

8 - APPLICATIONS DE LA GEODESIE

edgeoff	supprime les grains touchant le bord
fgrain	extrait le premier grain
border	contours internes du champ
clohole	bouche les pores internes
maxima, minima	extrema d'une fonction
extmaxima, extminima	extremas étendus
swamping	correction d'homotopie par marqueurs
extrema	extrema d’une image dans un masque
grainclose	lissage et fermeture de grains
indivaf	fermeture individuelle de particules
dynamics	dynamique des maxima et minima d’une image

9 - FILTRES

9.1. Filtres principaux

af	filtre alterné
asf, fullasf	filtre alterné séquentiel (2 types de progressions)
lineaf, lineasf	filtre alterné, alterné séquentiel sur lineopen
automed, centre	opérateur automédian, centre morpho.
contrast	contraste sur isoero-isodil, sur isopen-isoclose
contrasth	contraste sur top hat

9.2. Filtres supplémentaires

minifilt	mini filtre alterné
isaf, isasf	filtre alterné, alterné séquentiel, isotrope
buildaf, buildasf	filtre alterné, alterné séquentiel, par reconst.
closoropen	toggle entre ouverture et fermeture
minibuildaf	filtre par reconstruction avec miniopen et miniclose
binmiddle	ensemble médian binaire
grmiddle	ensemble médian numérique

10. BOULES MAXIMALES ET SQUELETTE

binopenskel	squelette par carrés (hexa.) maximaux
binultim	érodé ultime
centroid	centroïde (dernier érodé)
condbis	bissectrice conditionnelle

11 - AMINCISSEMENTS, EPAISSISSEMENTS

11.1. Amincissements, épaississements binaires

(gds)thickturn, (gds)thinturn	cycle d'épaississements, d'amincissements, élémentaires (géodésiques)
(gds)thick, (gds)thin	épaississement, amincissement, limite
Lthick, Mthick, Dthick	épaississements homotopiques
Lthin, Mthin, Dthin	amincissements homotopiques
endpoints	points extrèmaux
multpoints	points multiples
gdscentre	centre géodésique

11.2. Amincissements, épaississements numériques

greyseero, greysedil	érosion dilatation par se
greythickstep, greythinstep	épaississement, amincissement, élémentaires
dirgradient	gradient directionnel
vectgrad0	gradient complet brut
gradvect	gradient complet élaboré
mainthin	amincissement et ébarbulage
greyhmt	transformation en tout-ou-rien numérique
greythickturn	épaississement numérique élémentaire
greythick	épaississement numérique

12 - LIGNE DE PARTAGE DES EAUX

clip	ébarbulage

(gds) skiz	squelette (géodésique) par zones d'influence
threshwshed	L.P.E. par seuillage
. **mwshed, wshed**	L.P.E. marquée, ou non

13 - SEGMENTATION

mgradwshed, gradwshed	L.P.E. marquée ou non, du gradient
shapeseg	segmentation par fonction distance
smoothcnc, jumpcnc, jump	extraction de contraste
mosaic	image mosaïque
wfall	transformation hiérarchique (cascades)
kheops	pyramide d'images mosaïque

14 - MESURES

14.1. Mesures de base

diameter	variation diamétrale
digperim, perim	périmètres digital et euclidien
cnumber	nombre de connexité
binh(v)ferret	diamètre de Ferret

14.2. Mesures supplémentaires

var0, var1, var2	variances d'ordre 0, 1, 2
rug	rugosité
mse	différence quadratique moyenne
bincount	nombre de composantes connexes
flatcount	nombre de zones plates
flatzone	aire des zones plates

15 - COURBES

bincov	covariance binaire (directe ou rectangle)
vario1, vario2	variogrammes d'ordre 1 et 2
modcont	module de continuité
isogranul, granul	granulométries
covar	covariance
pvonl	mesure de l'érosion linéaire binaire
p1vonl	granulométrie linéaire binaire en nombre
p2vonl	granulométrie linéaire binaire en mesure

16 - SIMULATIONS ALEATOIRES

16.1. Simulations booléennes

point, points, regpoints	points aléatoires
isobool, isobool2	simulations d'ensembles booléens isotropes
elbool, tribool, dropbool	simulations ensembles booléens anisotropes
rose, hard, disjoint, flake	simulations de hiérarchies
conebool, conebool2	simulations de cônes booléens
rocky	fonds rocheux

16.2. Droites et partitions

lines, diags	droites poissonniennes
nestlines	hiérarchies de droites poissonniennes
steps	bandes poissonniennes

17. GRAPHES

Gero, Gdil	érosion, dilatations sur un graphe
Gopen, Gclose	ouverture, fermeture sur un graphe
setborder	marque les bords d'une image numérique
Gbuild	reconstruction sur un graphe planaire
label	génération d'une image label

18. UTILITAIRES 3D

Seqload	chargement d'une séquence
Seqsave	sauvegarde d'une séquence
Seqclr	effacement d'une séquence
Seqset	affecte une valeur donnée à une séquence
Seqcopy	copie une séquence
Seqinv	inverse une séquence
Seqadd	additionne toutes les images d'une séquence
Seqcadd	additionne une valeur constante à une séquence
Seqseqadd	addition de deux séquences
Seqcsub	soustraction d'une valeur constante à une séquence
Seqseqsub	soustraction de deux séquences
Seqcmul	multiplie une constante à une séquence
Seqmean	valeur moyenne des images d'une séquence
Seqseqmean	séquence moyenne de deux séquences
Seqdiff	module de la différence de deux séquences
Seqvolume	volume d'une séquence
Seqinf	inf de toutes les images d'une séquence
Seqseqinf	inf entre deux séquences
Seqsup	sup de toutes les images d'une séquence
Seqseqsup	sup de deux séquences
Seqbuild	reconstruit une séquence à l'aide d'une séq. marqueur
Seqthresh	seuillage à niveaux de gris d'une séquence
Seqgrad	gradient d'une séquence
Seqhist	histogramme d'une séquence
Scroll, Scroll2, Scroll3	balayage de séquence
Visu	visualisation en perspective

19. PROCEDURES 3D

ero3D, dil3D	érosion et dilatation 3D
open3D, Close3D	ouverture, fermeture 3D
gradient3D	gradient 3D
build3D	reconstruction 3D
buildopen3D, buildclose3D	ouverture, fermeture par reconstruction 3D
openth3D, **closeth3D**	chapeau haut-de-forme 3D

maxima3D, minima3D	extrema 3D
af3D, **asf3D**	filtres alternés séquentiels 3D
buildaf3D, **buildasf3D**	filtres alternés séquentiels par reconstruction 3D
timdil, timero	érosion, dilatation linéaire suivant l'axe temporel
timopen, **timclose**	ouverture, fermeture linéaire selon l'axe temporel

20. SOURIS

fill	dessine et remplit un polygone
dline	dessine des lignes
delobj	supprime des composantes connexes
pointinfo	coordonnées et valeur de gris d'un point
binpointer	pointe et extrait une particule
objinfo	aire d'un objet pointé
brush	fonction brosse