Sorbonne Université

LIP6

Vision

Opérateurs morphologiques

Règles de codages

Avec l'augmentation de la complexité des codes, la mise au point, le debug et la validation de ces codes va passer de facile à difficile à extrêmement complexe, si en amont certaines règles de codage et certains tests unitaires ne sont pas systématiquement fait. Ces règles sont les suivantes :

- Les accès mémoire doivent être fait via des macros : `load1(X,i)` pour accéder à `X[i]`, `load2(X,i,j)` pour accéder à `X[i][j]`, `store1(X,i,x)` pour écrire en `X[i]`, `store2(X,i,j, x)` pour écrire en `X[i][j]`.
- Tous les cas particuliers (dus au déroulage) des boucles doivent être traités. La fonction la plus complexe ayant un déroulage externe de 2 (`elu2`) et un détroulage interne de 3 `ilu3`, la taille des images doit donc varier de 3 en hauteur et 2 en largeur.
- Afin d'anticiper la mise au point des versions les plus complexes, les opérateurs morphologiques doivent aussi être utilisés à travers des macros. Par exemple `min3`, et `min2` (ou `ui8min3`, et `ui8min2` si ces macros font appel à des fonctions forcement plus typées). Durant la phase de mise au point, ces macros réaliseront des additions. De même, pour la dilatation, il faut des macros `max3`, et `max2`.
- le code de test (`morpho_test`) contient plus de code que nécessaire pour implémenter des opérateurs morphologiques binaire : faire attention si vous modifier cette partie.
- Enfin, la fonction principale appèle systématiquement une fonction ligne portant le meme nom qu'elle et préfixée par `line_`

1 Opérateurs d'érosion et de dilatation en niveau de gris

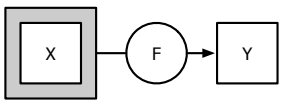

FIGURE 1 – Opérateur morphologique de rayon 1

Le but de cette étape pré-projet est d'implémeter plusieurs versions optimisées des opérateurs morphologiques d'érosion et de dilatation. Pour cela, il suffit de convertir les codes des stencils d'addition 3×3 manipulant des données de type `float` en des stencils d'érosion ou de dilatation manipulant des données de type `uint8`. Ce ne sont donc que quelques règles de réécriture à mettre en oeuvre dans un éditeur de code.

Opérateur d'érosion 3×3 en niveau de gris 8 bits (fichier `morpho_min3`) :

- `min3_ui8matrix_basic`
- `min3_ui8matrix_reg`
- `min3_ui8matrix_rot`
- `min3_ui8matrix_red`
- `min3_ui8matrix_ilu3`
- `min3_ui8matrix_ilu3_red`
- `min3_ui8matrix_elu2_red`
- `min3_ui8matrix_elu2_red_factor`
- `min3_ui8matrix_ilu3_elu2_red_factor`

Opérateur de dilatation 3×3 en niveau de gris 8 bits (fichier `morpho_max3`) :

- `max3_ui8matrix_basic`
- `max3_ui8matrix_reg`
- `max3_ui8matrix_rot`
- `max3_ui8matrix_red`
- `max3_ui8matrix_ilu3`
- `max3_ui8matrix_ilu3_red`
- `max3_ui8matrix_elu2_red`
- `max3_ui8matrix_elu2_red_factor`
- `max3_ui8matrix_ilu3_elu2_red_factor`

2 Opérateurs d'ouverture et de fermeture en niveau de gris

Cette étape s'intéresse à la composition d'opérateurs pour obtenir les opérateurs d'ouverture et de fermeture. On se concentrera uniquement sur l'ouverture, la fermeture s'obtenant en inversant les opérateurs de base (là aussi quelques règles de réécriture à mettre en ieuvre dans un éditeur).

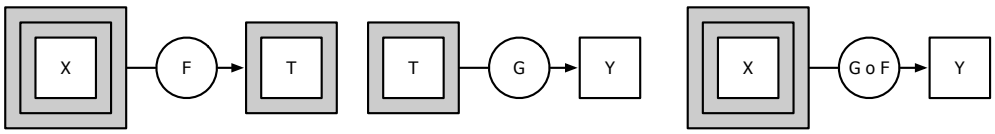

FIGURE 2 – Deux opérateurs en séquence vs deux opérateurs fusionnés

Version de référence :

- `ouverture3_ui8matrix_basic` : appel en séquence de `min3_ui8matrix_basic` et `max3_ui8matrix_basic`.

Dans toutes les versions, on fera attention à l'espace d'itération des boucles. Pour rappel, le plus simple est qu'une boucle parcours l'espace d'arrivé et non l'espace de départ. Dans un premier temps, il s'agira d'implémenter des versions pipeline : le premier opérateur produit les lignes une par une (sauf les versions `elu2` qui les produisent 2 par 2) qui seront consommées par à la même cadence par le second opérateur.

Versions pipeline :

- `ouverture3_ui8matrix_pipeline_red`
- `ouverture3_ui8matrix_pipeline_ilu3_red`
- `ouverture3_ui8matrix_pipeline_elu2_red`
- `ouverture3_ui8matrix_pipeline_elu2_red_factor`
- `ouverture3_ui8matrix_pipeline_ilu3_elu2_red`
- `ouverture3_ui8matrix_pipeline_ilu3_elu2_red_factor`

Dans un second temps, il s'agira d'implémenter des versions fusionnées : au lieu de passer par une mémoire temporaire T, l'ensemble des points sont consommés dans l'image source X, puis les deux opérateurs sont appliqués et le résultats est produit dans l'image destination Y.

- `ouverture3_ui8matrix_fusion`
- `ouverture3_ui8matrix_fusion_ilu_red`
- `ouverture3_ui8matrix_fusion_ilu_elu2_red`
- `ouverture3_ui8matrix_fusion_ilu_elu2_red_factor`

A noter que l'ordre de déroulage de la boucle externe n'est pas précisé. A vous de le trouver et de modifier le nom des fonctions en conséquence.