BESSAADI Ilyess

FRAPPART Théodore

Rapport de TP ATIV – Détection de contours

I] Introduction :

Le code source associé a ce rapport permet la compilation de notre rendu pour le TP de détection de contours. Pour le compiler ce programme, vous disposez également d'un fichier CMakeList, si vous ne disposez pas de l'utilitaire CMake, vous les trouverez ici.

Une fois compiler, le programme s'exécute à l'aide de lignes de commandes en fonction de l'utilisation que vous voudrez en faire.

Les arguments possibles sont les suivants :

« Chemin de l'image » « Chemin du fichier filtre » « Action » « indice du filtre à utiliser/Valeur du seuil haut » « Valeur du seuil bas ».

Chemin de l'image : Comme son nom l'indique, il s'agit de l'emplacement de l'image sur laquelle appliquer les filtres.

Chemin du fichier filter : Il s'agit de l'emplacement du fichier filtre a utilisé (Le dossier « ./data/filter/ » contient des fichiers préconçus).

Action : Spécifie l'action à appliquer sur l'image, il dispose de trois valeurs :

- filter : Cette action permet d'appliquer le filtre voulu à l'aide d'une convolution.
- ed2 : Cette action permet d'appliquer une détection de contour en x et y sur l'image choisie.
- ed4 : Cette action permet d'appliquer une détection de contour en x et y et sur les deux diagonales sur l'image choisie.

Indice du filtre à utiliser (optionnel) : Lors d'une action « filter », si votre fichier de chargement de filtre contient plusieurs filtres, cet argument permet de spécifier quel filtre utiliser.

Valeur du seuil haut : Lors d'une action « ed2 » ou « ed4 », cet argument permet de spécifier le seuil haut choisi (Une valeur réelle entre 0 et 1, 0 laissant passer toute valeur, 1 bloquant toutes valeurs inférieures à la valeur maximum trouvée dans l'image).

Valeur du seuil bas : Lors d'une action « ed2 » ou « ed4 », cet argument permet de spécifier le seuil bas choisi (Une valeur réelle entre 0 et 1, 0.5 par exemple nécessite que la valeur testée soit supérieure ou égale à 0.5 * la valeur nécessaire pour passer le seuil haut).

Exemples de ligne de commande valides :

./data/img/kirby.jpg ./data/filter/Kirsch_4D.txt ed4 0.4 0.7

./data/img/kirby.jpg ./data/filter/Prewitt_2D.txt ed2 0.8 0.3

./data/img/kirby.jpg ./data/filter/Blur3.txt filter

II] Details d'implémentation :

Le programme dispose d'une classe principale nommée Filter, qui gère le stockage (à l'aide de la classe cv ::Mat) et l'application des filtres.

La convolution est implémentée en suivant la formule disponible dans la description de la fonction OpenCV nommée filter2D, de plus, elle est capable de gérer toute taille de noyau (filtre) ne dépassant pas la taille de l'image sur laquelle s'applique la convolution (ce qui n'arrive normalement jamais).

Les calculs du gradient et de la pente pour la détection de contours en deux et quatre dimensions ont été implémentés en suivant les instructions vues en classe.

De même pour le seuillage par hystérésis et les maximas locaux.

Pour rendre l'utilisation de ce programme plus générique et pratique, nous avons aussi mis en place l'utilisation d'un parser de lignes de commande, permettant d'utiliser les différentes fonctionnalités du programme sans avoir à le recompiler.

Pour finir, nous avons aussi utiliser un système de fichier texte pour choisir les filtres à appliquer. Une base de filtres (Flou et flou gaussien 3*3 et 5*5, Prewitt, Sobel et Kirsch 2D et 4D) ainsi qu'un patron expliquant comment créer ses propres filtres (sample.txt) sont disponibles dans le fichier « ./data/filter ».

Par ailleurs, une fois que le programme a fini de s'exécuter, l'invité de commande affiche le temps total d'exécution du programme.