Le génie pour l'industrie

Département de genie logiciel et des TI

Rapport de Laboratoire

Numéro du laboratoire	3
Nom du laboratoire	Filtrage des images
Étudiant(s)	Eric Barry Axel Pajot
Code(s) permanent(s)	BARE01128901 PAJA07089006
Numéro d'équipe	
Cours	GTI410
Session	Automne 2017
Groupe	01
Chargé(e) de laboratoire	Roseline Olory Agomma
Date	09 Novembre 2017

Table des matières

Introduction

Le laboratoire consiste à ajouter des fonctionnalités de filtrage à l'application afin de pouvoir utiliser un filtrage choisi. Les options de filtres à implémenter sont les filtres moyen, Gaussien, Sobel et Laplacien. L'application devra être en mesure d'appliquer les filtres en gérant les bordures de l'image entrée selon une méthode choisie. La réalisation du laboratoire demande une analyse du système existant et de son fonctionnement. Il faudra, par la suite, modifier les classes existantes pour faire l'intégration de la classe personnalisée liée au filtre, pour créer les matrices de filtrage et pour implémenter la stratégie de gestion de bordure et de conversion de valeurs de pixels. Il faudra finalement tester l'application avec différentes images et différents ensembles de résultats afin de s'assurer que l'application fonctionne bien.

Analyse

Les fonctionnalités à ajouter dans l'application ont pour but de permettre d'appliquer un filtre sur une image et d'expérimenter divers types de filtrage selon le choix de l'utilisateur.

Le programme est lancé puis une image sur laquelle appliquer les modifications est choisie. Ensuite un choix de bordure, de stratégie et de filtre à appliquer doit être fait. Ces informations sont prises par les classes de filtre puis un convertisseur utilise un algorithme pour transformer cette image dans un format plus facile à manipuler. L'image résultante est alors modifiée pour lui ajouter une bordure fine qui servira pour l'application du filtre. Lorsque la bordure est appliquée, le filtre ayant été choisi est appliqué par convolution sur l'image. Le programme affiche alors l'image résultante à l'utilisateur.

Conception

Une classe de filtre personnalisé “LabFilter” gère le choix du filtre selon ce qui a été passé par les classes FilterKernelPanel, KernelPanel et FilteringTransformer.
Des matrices ont été modifiées dans FilterKernelPanel afin de permettre d’avoir les bonnes valeurs lors de leurs sélections dans l’interface, ces valeurs sont ensuite récupérées pour charger la matrice utilisée pour effectuer le produit de convolution dans la classe LabFilter.

La méthode mouseclicked de FilteringTransformer a étée modifiée afin de permettre le lancement des opérations de conversion et d’application de filtre.

Une classe ImageClampAbsTo255Strategy qui hérite de ImageConvertionStrategy a été créée afin de pouvoir sélectionner et appliquer cette nouvelle conversion dans l’application. Elle permet de prendre la valeur absolue des pixels négatifs et de réduire à 255 la valeur maximale des pixels.

La classe borderManager a étée ajoutée afin de gérer le traitement de la bordure. Celle-ci parcourt les pixels de l’image afin de la reproduire avec une bordure ajoutée.

La classe LabFilter est la classe qui gère le procédé de filtrage, en suivant les paramètres entrés par l’utilisateur.

Implémentation et algorithmes

classe BorderManager

```
ManageBorder( typeDeBordure)

si c'est typeDeBorduer == "copie"
        Pour la largeur i
                Pour la hauteur j
                        tant que i > 0 et i < largeur et que j > 0 et j < hauteur
                                imageBordee.setPixel(i,j) = imageOrigine.getPixel(i-1, j-1)

        Pour la largeur i
                Pour la hauteur j
                        si (i == 0) && (j == 0 )
                                imageBordee.setPixel(i,j) = imageOrigine.getPixel(i, j)

                                si ((i == 0) && (j == hauteur - 1))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i, j-2)

                                si ((i == largeur -1) && (j == hauteur ))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i-2, j-2)

                                si ((i == largeur  -1) && (j == 0))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i-2, j)

                                si ((i == 0) && (j > 0 && j < hauteur ))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i, j-1)

                                si ((i > 0 && i < width-1) && (j == 0))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i, j-1)

                                si ((i > 0 && i < width-1) && (j == hauteur ))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i-1, j-2)

                                si ((i == largeur -1) && (j > 0 && j < hauteur-1))
                                        imageBordee.setPixel(i,j) = imageOrigine.getPixel(i-2, j-1)

        retourne imageBordee
```

Stratégie pour convertir l'ImageDouble et ImageX (valeur absolue et normalisation vers 255)

Pour x de 0 a largeur de l'image

```
Pour y de 0 a la hauteur de l'image

        pixelBleu = valeur absolue de pixelBleu
        pixelVert = valeur absolue de pixelVert
        pixelRouge = valeur absolue de pixelRouge
        Valeur max = valeur max entre pixelBleu pixelVert et PixelRouge

        Si valeurMax superieur a 255 alors
            pixelBleu =( pixelBleu / valeurmax) * 255
            pixelVert =( pixelVert / valeurmax) * 255
            pixelRouge =( pixelRouge / valeurmax) * 255
        newImage.setPixel (pixelBleu,pixelvert,pixelRouge)
```

Discussion

La méthode choisie pour l'ajout de bordure de l'image est la méthode par copie. La méthode est simple à faire à la main, mais demande plus de réflexion pour son implémentation. Ainsi beaucoup de temps a été investi pour réaliser la méthode qui ajoute la bordure sur l'image. Il y a probablement une meilleure façon de faire le système de copie de pixels pour la gestion de la bordure.

Points faibles:

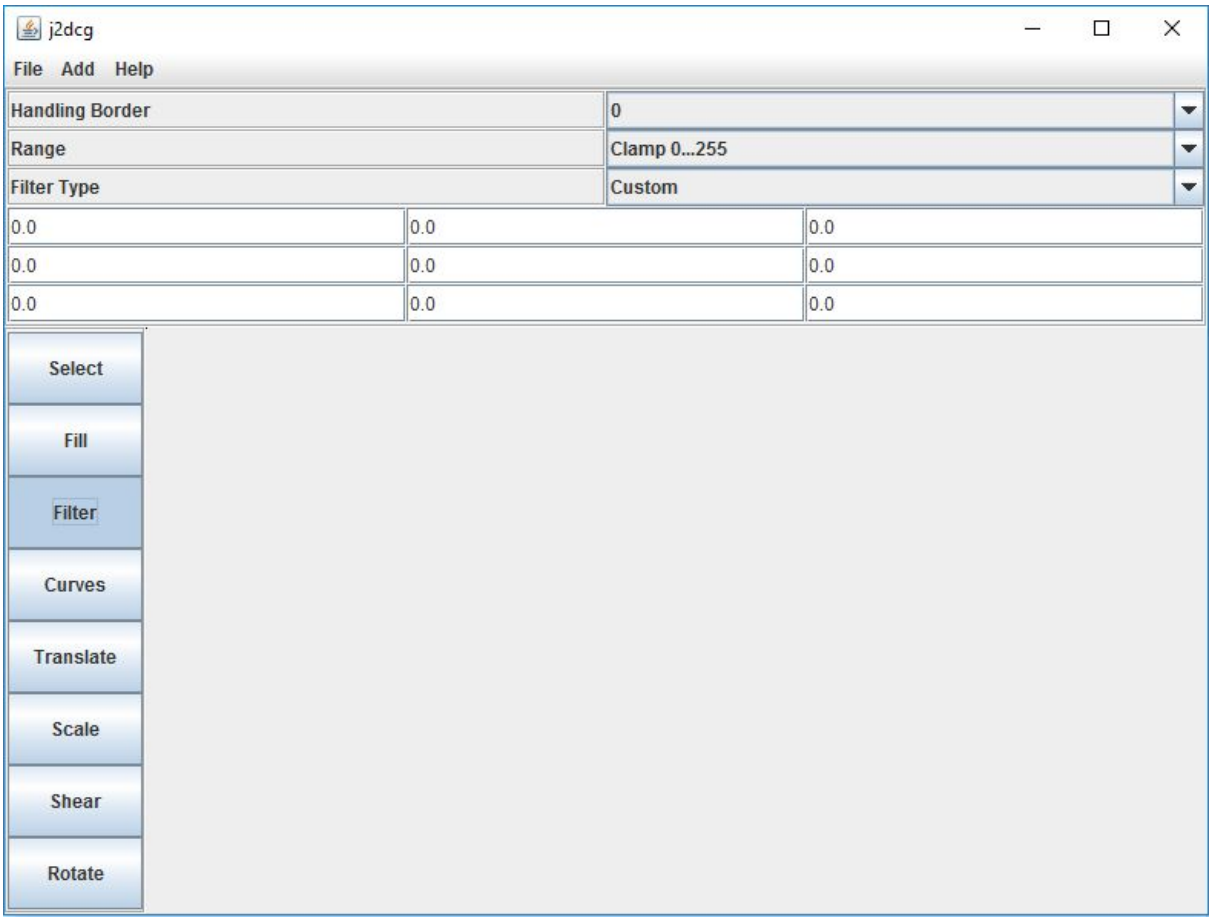
- L'algorithme qui génère la bordure pourrait être amélioré, peut être il y a une librairie qui pourrait remplacer cet algorithme
- Cela aurait été intéressant d'ajouter un zoom sur l'image entrée dans le programme, pour permettre de mieux voir les résultats d'un filtrage.
- Il serait intéressant d'ajouter un système redo/undo pour les manipulations
- Il serait intéressant d'ajouter un écran qui affiche les modifications ayant été faites à l'image source, et de pouvoir revenir à cet état.

Manuel de l’utilisateur

Le programme peut être exécuté en lançant le java -jar LAB03.jar, l’utilisateur doit avoir un JRE d’installé sur sa machine pour exécuter le programme.

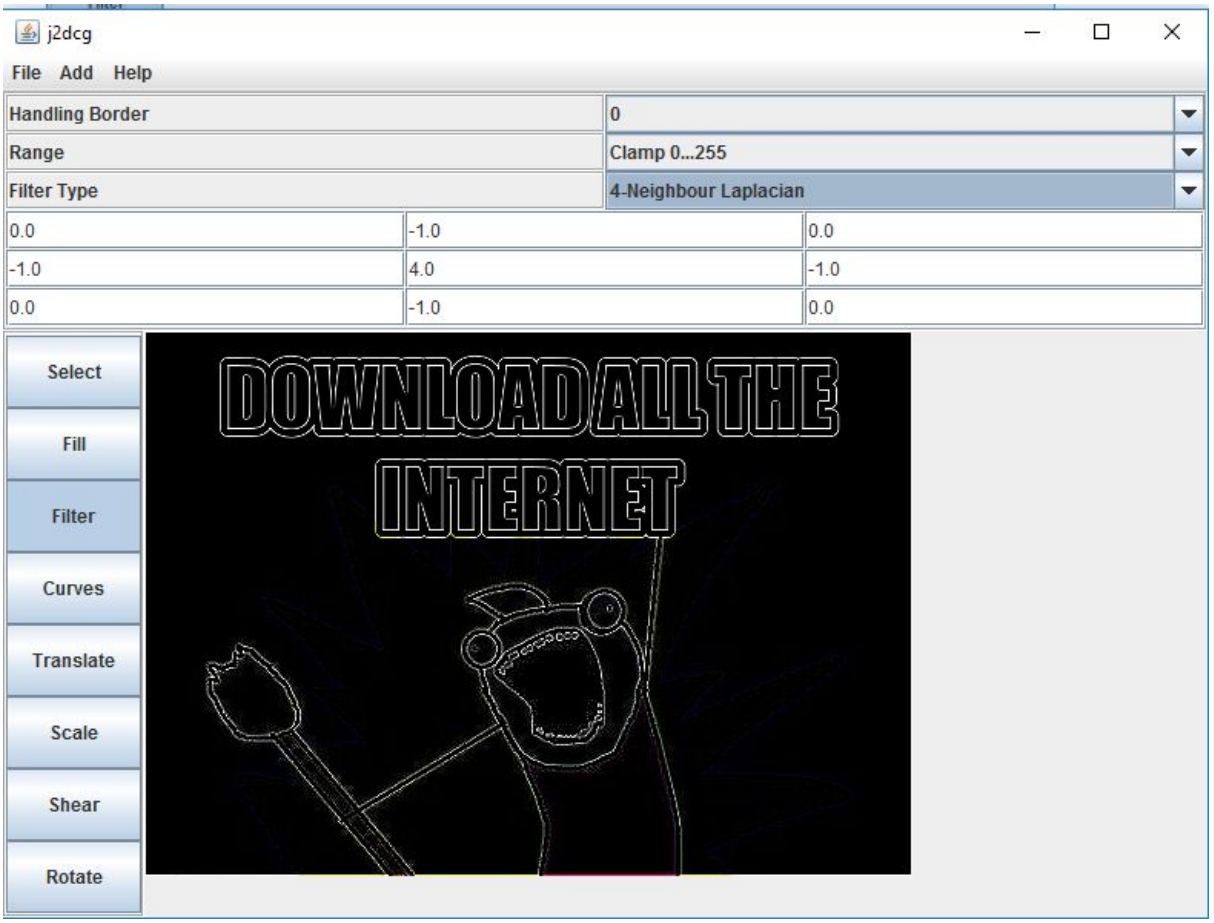
Le programme s’ouvre et il est possible de voir la fenêtre principale. Pour utiliser les fonctions de remplissage, il faut aller dans l’onglet “Filter”.

Figure 1.1 : Fenêtre principale



Dans l’onglet filter l’utilisateur a accès aux option suivantes :
-La gestion des bordures par copie, par miroire ou circulaire.
-Le range des couleurs des pixels
-Le type de filtre : mean, Gaussian, Laplacian, Prewitt, Sobel.

L’utilisateur peut aussi remplire directement le filtre voulue dans cases du tableau.



Exemple d'utilisation, ici la gestion de la bordure est faite par l'utilisation de 0 dans les cases extérieures, le range des pixels est restreint entre 0 et 255. Le filtre choisi est le filtre Laplacian qui permet la détection des contours. On peut observer que sur le résultat les contours sont bien dessinés.

Conclusion

Le laboratoire permet d'appliquer des filtres avec différentes options sur une image. La réalisation du laboratoire a permis aux membres de l'équipe de mieux comprendre comment sont appliqués les filtres par convolution et de visualiser leurs effets sur une image.

L'analyse du travail s'est faite en examinant le code et en testant différentes alternatives pour la réalisation du travail. Le design et l'implémentation du travail ont été fait en testant différentes façons de faire dans le code et en choisissant la méthode d'ajout de bordure par copie et la conversion imageX/ImageDouble est faite avec la valeur absolue des valeurs des pixels ainsi que la normalisation à 255.
L'application permet à l'utilisateur de choisir dynamiquement les options voulus pour le filtrage de l'image et ainsi de lui permettre de tester et évaluer les différents résultats.
Suivant la réussite de ce laboratoire, l'équipe est prête à aborder le prochain.