Cours de programmation séquentielle

Transformations géométriques et filtres (suite)

1 Les matrices

1.1 Buts

— Implémentation de convolutions.
— Implémentation de transformations géométriques sur des images (translation, zoom).
— Utilisation de tests unitaires.

1.2 Énoncé

Il s'agit ici de compléter la librairie de traitement d'images que vous avez commencé depuis quelques semaines. Dans un premier temps implémenter des transformations géométriques (translations, zoom), puis implémenter les filtres pour le traitement d'images.

1.3 Cahier des charges

1.3.1 Transformations géométriques

Les fonctions pour les transformations géométriques sont à mettre dans des fichiers `geom.h` et `geom.c` respectivement pour les entêtes et leur implémentation. Il faudra donc:

— implémentation d'une fonction de translation d'une matrice `mat` à l'aide d'une matrice de translation `translation` qui retourne la matrice translatée, ainsi que la version "en place", qui retourne `1` si tout s'est bien passé, `0` sinon. La matrice `translation` est connue et est 2×1 (c'est un vecteur bi-dimensionnel).

```
matrix geom_translate(matrix mat, matrix translation);
int geom_translate_in_place(matrix *mat, matrix translation);
```

— implémentation d'une fonction de de zoom d'une matrice `mat` à l'aide d'une matrice de translation `translation` qui retourne la matrice translatée, ainsi que la version "en place", qui retourne `1` si tout s'est bien passé, `0` sinon. La matrice `factors` est un vecteur bidimensionnel (de taille 2×1) et contient les facteurs de zoom dans chaque direction de l'espace.

```
matrix geom_zoom(matrix mat, matrix factors);
int matrix_geom_zoom_in_place(matrix *mat, matrix factors);
```

1.3.2 Transformations de matrices

Les fonctions pour les transformations de matrices sont à mettre dans les fichiers `matrix.h` et `matrix.c`. Il faudra donc:

— implémentation d'une fonction (et sa contrepartie "en place") qui permet de "clipper" les valeurs d'une matrice entre `0` et `1`. Tout ce qui est plus petit que `0` est ramené à `0` et tout ce qui dépasse `1` est ramené à `1`. Our plus d'informations sur clipper voir https://en.wikipedia.org/wiki/Clipper_(electronics).

```
matrix matrix_clipper(matrix mat);
int matrix_clipper_in_place(matrix mat);
```

— implémentation d'une fonction (et sa contrepartie "en place") qui permet de calculer une convolution avec un noyau de convolution, `kernel`. Lors de l'application du `kernel`, certaines valeurs sur les "bords" de la matrices `mat` sont inconnues. Pour ces valeurs-là il faut prendre `0`.

```
matrix matrix_convolve(matrix mat, matrix kernel);
int matrix_convolve_in_place(matrix *mat, matrix kernel);
```

1.3.3 Filtres

Les fonctions de filtrage de matrices sont à implémenter dans les fichiers `filter.h` et `filter.c`. Il faudra donc implémenter les fonctions de filtrage suivantes (comme d'habitude les fonctions "en place" retourne `1` si tout s'est bien passé, `0` sinon).

```
matrix filter_sharpen(matrix mat);
int filter_sharpen_in_place(matrix *mat);
matrix filter_blur(matrix mat);
int filter_blur_in_place(matrix *mat);
matrix filter_edge_enhance(matrix mat);
int filter_edge_enhance_in_place(matrix *mat);
matrix filter_edge_detect(matrix mat);
int filter_edge_detect_in_place(matrix *mat);
matrix filter_emboss(matrix mat);
int filter_emboss_in_place(matrix *mat);
```

Les matrices de convolution de tous ces filtres se trouvent à l'adresse https://docs.gimp.org/2.8/en/plug-in-convmatrix.html. Nous les reproduisons ici:

$$\underline{\underline{K}}_{\text{sharpen}} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \tag{1}$$

$$\underline{\underline{K}}_{\text{blur}} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \tag{2}$$

$$\underline{\underline{K}}_{\text{edge enhance}} = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{3}$$

$$\underline{\underline{K}}_{\text{emboss}} = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}. \tag{4}$$

Le filtrage s'effectue en trois étapes:

1. Normalisation de la matrice.
2. Convolution avec le noyau de convolution approprié.
3. Application du clipper.

1.4 Indications

— Pour la translation et le zoom, il faut bien penser à allouer (réallouer) une matrice "cible" plus grande/petite, car les pixels qui ne sont pas dans la matrice sont perdus.
— Pour le zoom, refléchissez bien à comment combler les "trous". Une possibilité est de choisir la valeur du pixel la plus proche de l'image d'origine. Pour ceux que ça intéresse, une façon plus jolie de procéder se trouve là https://en.wikipedia.org/wiki/Bilinear_interpolation.
— La translation par un nombre non-entier, va créer des "trous". Il faut donc réfléchir à comment faire pour qu'ils ne soient pas présents.

1.5 Bonus

Implémenter une fonction `crop` qui permet de ne garder qu'une partie rectangulaire d'une image d'origine.