INFO0030 : Projet de Programmation
Traitement d'Images par Application de Filtres

B. Donnet, E. Marechal

1 Contexte

Une image peut être décomposée en un tableau de points élémentaires appelés *pixels* (abréviation de *picture element*). Supposons que nous ne manipulons que des images en niveaux de gris (ou *grayscale* en anglais), i.e., des images dont les "couleurs" sont uniquement des nuances de gris. On peut représenter une telle image par une matrice d'entiers, dont la valeur des éléments représente l'intensité lumineuse des pixels de l'image. Par conséquent, un traitement d'image peut être réalisé en manipulant la matrice qui la représente.

Pour ce projet, nous vous demandons d'écrire un programme qui permet d'appliquer un filtre graphique particulier à une image qui vous est donnée dans un fichier au format `PNM` (*Portable Anymap*). Ce format est bien connu et a fait l'objet d'un projet précédent. Il vous est donc demandé, dans ce projet, de réutiliser votre librairie permettant de manipuler des images `PNM`.

2 Filtres

Dans cette section, nous décrivons les différents filtres que vous devrez appliquer sur des images `PNM`.

2.1 Retournement

Le filtre de *retournement* effectue une rotation de 180°de l'image. Dit autrement, ce filtre produit une symétrie axiale d'axe horizontal (i.e., ce qui est en bas va en haut et vice-versa). La Fig. 1(b) illustre l'application du filtre de retournement sur la Fig. 1(a).

Le filtre de retournement s'applique à n'importe quelle image (i.e., `PBM`, `PGM`, `PPM`).

2.2 Monochrome

Le filtre *monochrome* permet d'obtenir une version monochrome d'une image couleur.

Pour obtenir une version monochrome d'une image, il suffit de transformer chaque couleur de pixel $c = (r, v, b)$ [1] en annulant une des trois composantes, i.e.,

- $c' = (r, 0, 0)$, pour un filtre monochrome rouge
- $c' = (0, v, 0)$, pour un filtre monochrome vert
- $c' = (0, 0, b)$, pour un filtre monochorme bleu

La Fig. 1(c) illustre l'application d'un filtre monochrome rouge sur la Fig. 1(a).

Le filtre monochrome ne s'applique qu'aux images `PPM`.

2.3 Négatif

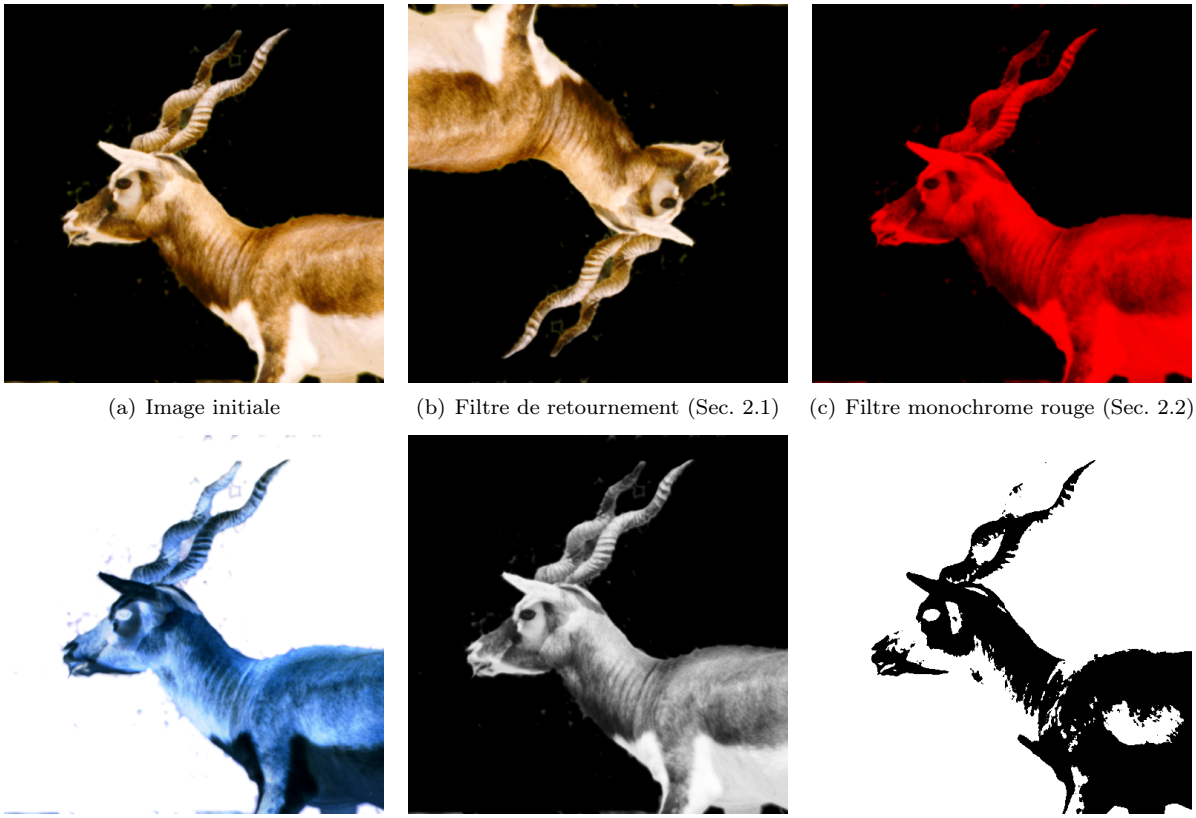
Le filtre *négatif* permet d'obtenir le négatif d'une image couleur.

La couleur négative d'une couleur $c = (r, v, b)$ est la couleur $c' = (255 - r, 255 - v, 255 - b)$.

La Fig. 1(d) illustre l'application d'un filtre monochrome rouge sur la Fig. 1(a).

Le filtre négatif ne s'applique qu'aux images `PPM`.

1. Le triplet (r, v, b) désigne l'intensité de rouge (r), de vert (v) et de bleu (b) d'un pixel.



(a) Image initiale (b) Filtre de retournement (Sec. 2.1) (c) Filtre monochrome rouge (Sec. 2.2)

(d) Filtre négatif (Sec. 2.3) (e) Filtre gris (Sec. 2.4) (f) Filtre NB (Sec. 2.5)

FIGURE 1 – Application des différents filtres.

2.4 50 Nuances de Gris

Le filtre *gris* permet de construire une version nuances de gris d'images en couleurs. Une nuance de gris est une couleur obtenue en donnant la même valeur aux trois composantes (rouge, vert, bleu) d'une couleur donnée.

Nous vous proposons deux techniques pour l'application du filtre gris :

1. on attribue la moyenne arithmétique des trois composantes (rouge, vert, bleu) à une nouvelle couleur, i.e., $c' = int(round(\frac{r+v+b}{3}))$.
2. on applique la formule suivante : $c = int(round(0.299 \times r + 0.587 \times v + 0.114 \times b))$.

La fonction $round()$ arrondi au nombre le plus proche, tandis que la fonction $int()$ retourne la valeur entière associée à un nombre réel.

La Fig. 1(e) illustre l'application d'un filtre gris (deuxième technique) sur la Fig. 1(a).

Le filtre gris ne s'applique qu'aux images `PPM` et produit une image `PGM`.

2.5 Noir & Blanc

Le filtre *NB* permet de construire une version en noir et blanc seulement (pas de gris intermédiaire) d'images en couleurs.

Le principe consiste à appliquer d'abord le filtre gris (cfr. Sec. 2.4) et transformer, ensuite, ce gris en un noir ou un blanc selon que la nuance de gris est ou supérieure à un seuil fixé.

La Fig. 1(f) illustre l'application d'un filtre NB (avec une valeur de seuil de 115 – le filtre gris a été appliqué avec la deuxième formule) sur la Fig. 1(a).

Le filtre NB s'applique aux images `PPM` et `PGM` et produit une image `PBM`.

3 Enoncé du Projet

Il vous est demandé d'écrire un programme C permettant d'appliquer sur des images `PNM` les filtres expliqués à la Sec. 2.

Votre projet devra :

— être soumis dans une archive `tar.gz`, appelée `filtres.tar.gz`, via la plateforme de soumission. [2] La décompression de votre archive devra fournir tous les fichiers nécessaires dans le répertoire courant où se situe l'archive. Vous penserez à joindre tous les codes sources nécessaires.

— réutiliser la librairie PNM que vous avez réalisé dans un précédent projet. Il est évident que l'équipe pédagogique s'attend à ce que vous ayez apporté des modifications à votre librairie PNM en fonction du feedback donné. A cette fin, vous nous fournirez le code source nécessaire de votre librairie PNM.

— être modulaire, i.e., nous nous attendons à trouver un (ou plusieurs) header(s) et un (ou plusieurs) module(s). Il est évident que votre projet doit contenir un programme utilisant le code écrit.

— appliquer les principes de la programmation défensive (vérification des préconditions, vérification des mallocs, ...). Pensez à libérer la mémoire allouée en cours de programmation afin d'éviter les fuites de mémoire.

— être parfaitement documenté. Vous veillerez à documenter correctement chaque header/fonction/-procédure/structure de données que vous définirez. Votre document suivra les principes de l'outil `doxygen`. [3]

— implémenter les fonctions permettant de filtrer une image (cfr. Sec. 2).

— être validé par une librairie de tests unitaires. Pour cela, vous utiliserez l'outil `seatest` vu au cours. [4] N'oubliez pas de joindre, dans votre archive, les fichiers `seatest.c` et `seatest.h`. Ces tests unitaires porteront sur votre librairie `PNM` mais aussi sur certaines fonctionnalités que vous aurez mis en place pour ce projet (à vous de voir lesquelles sont les plus pertinentes).

— comprendre un `Makefile` permettant au moins de

1. compiler vos tests unitaires pour votre librairie `PNM`. L'exécution de la commande

```
$>make pnm_tests
```

   doit produire un fichier binaire, appelé `pnm_tests`, permettant de tester votre librairie `PNM`. Ce fichier binaire devra pouvoir être exécuté de la façon suivante :

```
$>./pnm_tests
```

2. compiler votre projet. L'exécution de la commande

```
$>make filtre
```

   doit produire un fichier binaire, exécutable, appelé `filtre`. Attention, lors de la compilation, vous ne pouvez pas recompiler les fichiers sources propres à la librairie `PNM`. A la place, vous devez intégrer la librairie `libpnm` (cfr. projet précédent). Par contre, vous devez fournir les fichiers sources pour cette librairie dans votre archive `tar.gz` afin que l'équipe pédagogique puisse vérifier que vous avez pris en compte le feedback donné. le fichier binaire généré, `filtre`, doit pouvoir être exécuté de la façon suivante :

```
$>./filtre -i <image_input> -f <filtre> [-p <parametre>] -o <image_output>
```

   où `<image_input>` est l'image au format `PNM` qu'il faudra manipuler dans le programme, `<filtre>` le filtre à appliquer et `<param>` l'éventuel paramètre associé au filtre. Les valeurs possibles pour l'option `<filtre>` sont les suivantes :

   retournement Applique le filtre de retournement (cfr. Sec. 2.1). Il n'y a pas de paramètre associé à ce filtre (d'où le côté optionnel de l'option `-p`).

2. Pour rappel, l'adresse de la plateforme de soumission est `https://submit.montefiore.ulg.ac.be`
3. Vous veillerez, aussi, à documenter en `doxygen` votre librairie `PNM`.
4. Le code source de `seatest` est disponible sur la page Web du cours.

monochrome Applique le filtre monochrome (cfr. Sec. 2.2). Le paramètre associé peut être soit `r` (application d'un filtre monochrome rouge), `v` (application d'un filtre monochrome vert), ou `b` (application d'un filtre monochrome bleu).

negatif Applique le filtre négatif (cfr. Sec. 2.3). Il n'y a pas de parmaètre associé à ce filtre.

gris Applique le filtre gris (cfr. Sec. 2.4). Dans ce cas-çi, le paramètre sera soit 1 (application de la moyenne arithmétique), soit 2 (application de la deuxième formule).

NB Applique le filtre NB (cfr. Sec. 2.5). Dans ce cas-çi, le paramètre sera le seuil (une valeur comprise entre 0 et 255).

Le résultat du filtrage sera placé dans un nouveau fichier `PNM` dont le nom sera indiqué par l'option `-o <image_output>`.

3. générer de la documentation. L'exécution de la commande

```
$>make doc
```

doit produire une documentation, au format HTML, dans le sous-répertoire `doc/`.

Pour tester votre code, vous pouvez réutiliser les fichiers `PGM` donnés lors du premier projet en plus des images présentées dans cet énoncé (Fig. 1 – `antilope.ppm`, `antilope_gris.pgm`, `antilope_monochrome.ppm`, `antilope_nb.pbm`, `antilope_negatif.ppm` et `antilope_retournement.ppm`)

Il est impératif de respecter **scrupuleusement** les consignes sous peine de se voir attribuer une note de 0/20 pour non respect de l'énoncé.