Mini-Projet C++ **RIBER Rafael**
ColoReduce
ICC Pratique - CS-119
Novembre - Décembre 2018

Phase d'analyse

Decrire l'organisation generale du programme en faisant ressortir la mise en oeuvre des principes d'abstraction et de re-utilisation.

Le programme est divisé en 5 fonctions de lecture regroupées sous `fileRead()`, 4 fonctions de transformation, et 2 fonctions de rendu:

Au sein de la fonction `fileRead`, executée dans `main` et qui regroupe les autres fonctions de lecture du fichier d'entrée, on crée tout d'abord une strucure `"image"` de type `InputImg` qui stockera toutes les informations du fichier d'entrée nécessaires dans des champs facilement accessibles. On execute ensuite dans `fileRead` les fonctions `inputReduced, inputThresholds, inputFilters, inputDimensions` et `inputPixels` qui remplissent chacune un champ de cette structure qui leur est passée par référence, respectivement le nombre et les valeurs des couleurs réduites, les valeurs des seuils, le nombre de filtres, les dimensions `nbL` et `nbC`, et les différentes couleurs des pixels de l'image à traiter, tout en vérifiant la validité de ces données et en renvoyant des erreurs en cas d'une mauvaise valeur. Le résultat de cette lecture est stocké dans la structure de type `InputImg` appelée `image`.

Cette structure est ensuite passée à la première fonction de transformation, appelée `normalize`, qui crée un tableau de type `NormImg` appelé `norm`, et calcule

Algorithme de filtrage

Algorithme 1 : FILTRAGE

Entrées : Image normalisée N de taille $C \times L$, nombre de filtres à appliquer f, nombre de couleurs réduites de l'image r.
Résultat : Image N filtrée f fois
Remarque: Les coordonnées de l'image ainsi que les listes sont ici indexées à 0.

```
1  copie ← N
2  val ← 0
3  count est une liste de longueur fixe r.
4  Pour n de 0 à f
5      Pour x de 1 à C − 1
6          Pour y de 1 à L − 1
7              current ← 0
8              Pour i de −1 à 1
9                  Pour j de −1 à 1
10                     Si i ≠ 0 ou j ≠ 0
11                         current ← copy[x + i][y + j]
12                         Pour c de 0 à r
13                             Si c = current
14                                 count[c] = count[c]+1
15                                 Si count[c] ≥ 6
16                                     val ← c
17                                     Continuer à la ligne 20.
18                                 Sinon
19                                     val ← 0
20             N[x][y] ← val
21     copie ← N
22 Si f > 0
23     Pour i de 0 à L
24         Pour j de 0 à C
25             Si i = 0 ou j = 0 ou i = L − 1 ou j = C − 1
26                 N[i][j] ← 0
27 Sortir N
```

Analyse de complexité

Dans le pire des cas, on parcourt tous les pixels n'étant pas en bordure de l'image f fois, en parcourant une liste de longueur r pour chacun des 8 voisins d'un pixel, puis les pixels de bordure une seule fois.

On a donc $N_i = (c-2)(l-2)$ pixels à filtrer f fois, et $N_b = (2c+2(l-2))$ pixels en bordure, le tout multiplié par le nombre maximum de couleurs réduites $r_{max} = 255$ que l'on parcourt 8

fois pour les pixels voisins, ce qui donne:

$$N_{pixels} = 8 \cdot r_{max}\left(f \cdot N_i + N_b\right) = \underbrace{8 \cdot r_{max} \underbrace{\left(flc - f \cdot \left(2c + 2l - 4 - \frac{2c}{f}\right)\right)}_{\leq f \cdot l \cdot c}}_{\leq r_{max} \cdot f \cdot l \cdot c} \leq 8 \cdot rflc$$

Et donc:

$$N_{pixels} \leq 8 \cdot rflc$$

On a ainsi une complexité de $\mathcal{O}(N_{pixels}) \leq \mathcal{O}(8 \cdot r \cdot f \cdot l \cdot c)$, et donc une complexité de type $\mathcal{O}(??????????????????)$.