Structures de données et algorithmes

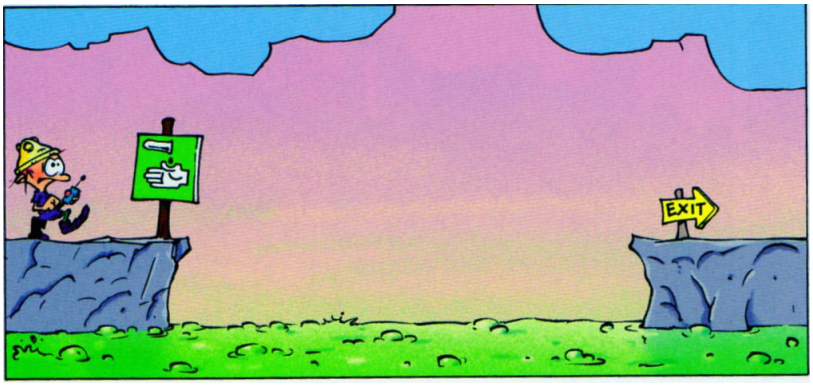
Projet 3

Mise en page automatique d'une bande dessinée

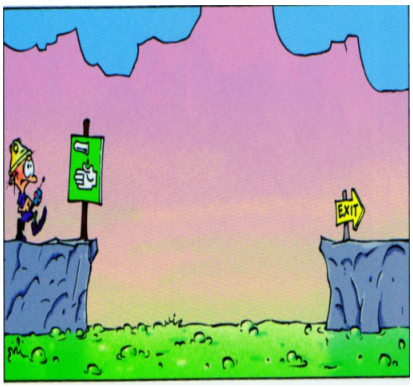
Jean-Michel Begon– Romain Mormont – Pascal Fontaine

4 mai 2020

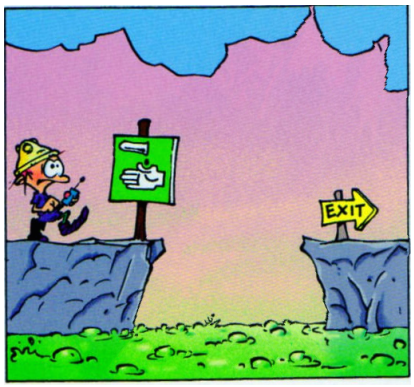
L'objectif du projet est d'implémenter un algorithme permettant de réduire la largeur des images sans modifier l'aspect des éléments qui la constituent afin d'éviter les effets d'écrasement. Pour ce faire, nous allons développer un algorithme de programmation dynamique qui va enlever des éléments non intéressant de l'image (figure 1).



(a) Image originale.



(b) Image écrasée horizontalement.



(c) Image sans effet d'écrasement.

Figure 1 – Réduction de largeur de moitié. Réduire naïvement la largeur de l'image sans toucher à la hauteur crée un effet d'écrasement (1b). On peut éviter celui-ci au prix de certains détails (1c).

1 Redimensionnement des images

Soit une image I représentée par un tableau de taille $m \times n$. $I[i, j]$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$ est le pixel de la $i$ème ligne (en commençant à compter en haut de l'image et vers le bas) et de la $j$ème colonne (en commençant à compter à gauche et vers la droite).

Le redimensionnement d'image consiste à faire passer une image de sa résolution initiale $n \times m$ à une résolution cible $n' \times m'$ différente. Ici, on se focalisera sur les réductions de la largeur uniquement $(n' = n, m' < m)$.

L'algorithme de redimensionnement que nous allons développer consiste à supprimer un pixel de chaque ligne de manière à faire diminuer la largeur d'une colonne. Ce procédé est répété $k = m - m'$ fois pour obtenir la largeur finale de l'image.

Chaque réduction doit respecter certains critères afin d'obtenir une image pertinente. En particulier,

1. le pixel supprimé à la ligne $i + 1$ doit être "voisin" de celui supprimé à la ligne i, afin de ne pas créer d'artefacts de décalage ;
2. les pixels sélectionnés doivent être les moins "importants" possibles, afin de ne pas perdre des détails pertinents de l'image.

Sillon Nous considérerons deux pixels (i, j) et (i', j') comme voisin si $|i - i'| \leq 1$ et $|j - j'| \leq 1$.

Un sillon (vertical) s dans une image de taille $n \times m$ est défini comme une suite ordonnée de pixels $(1, j_1), (2, j_2), \ldots, (n, j_n)$ telle que $j_k \in \{1, \ldots, m\}$ pour tout $1 \leq k \leq n$ et $|j_k - j_{k-1}| \leq 1$ pour tout $2 \leq k \leq n$. Un sillon connecte donc le haut et la bas de l'image au travers d'un chemin de pixels voisins.

Importance On évaluera l'importance du pixel (i, j) par son énergie, donné par

$$E(i, j) = E(i, j, \mathrm{R}) + E(i, j, \mathrm{G}) + E(i, j, \mathrm{B}) \tag{1}$$

$$E(i, j, c) = \frac{|I(i-1, j, c) - I(i+1, j, c)|}{2} + \frac{|I(i, j-1, c) - I(i, j+1, c)|}{2} \tag{2}$$

avec $I(i, j, c)$ la valeur associée au pixel (i, j) dans le canal de couleur $c \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ de l'image. Pour les pixels au bord de l'image, la ou les valeurs de pixels inexistantes seront remplacées dans la formule par la valeur $I(i, j, c)$ du pixel lui-même.

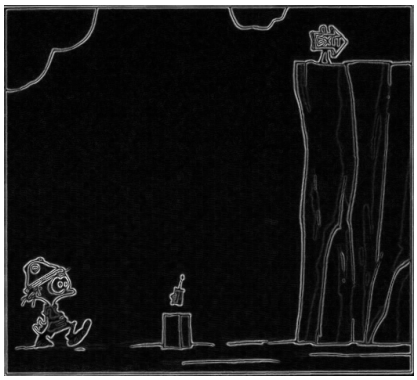
Intuitivement, E mesure à quel point un pixel est proche de ses quatre principaux voisins. S'il est similaire, on peut le supprimer sans craindre d'altérer un détail important de l'image (voir figure 2a).

Redimensionnement – sillon d'énergie minimale Dans l'optique de réduire la largeur d'une image d'un pixel (par ligne), on cherchera à déterminer le sillon s^* d'énergie minimale, l'énergie d'un sillon étant définie comme la somme de l'énergie des pixels qui la composent.

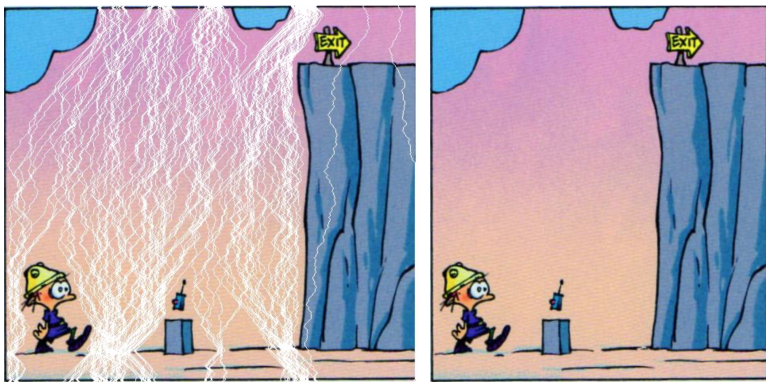
On peut déterminer le sillon d'énergie minimale par programmation dynamique. Pour ce faire, il faut formuler récursivement la fonction de coût $C(i, j)$, l'énergie du sillon d'énergie minimale s'arrêtant au pixel (i, j). Sur base de cette fonction, on peut déterminer le sillon optimal en comparant tous ceux s'arrêtant sur la dernière ligne de l'image.

La figure 2b illustre la mise en œuvre de l'algorithme. On note que les sillons d'énergie évitent les éléments importants, ainsi que les zones de changement brusque de couleur.

Implémentation rapide Une approche naïve consiste appliquer k fois la suppression du sillon d'énergie minimale. Néanmoins, entre deux suppressions successives la valeur d'énergie de la plupart des pixels n'est pas modifiée. Une implémentation efficace évitera le re-calcul inutile de ces valeurs.



(a) Energie de l'image. Noir signifie peu d'énergie.



(b) A gauche, l'image de départ où les 100 sillons d'énergie minimale sont représentés en blanc. A droite, la version réduite de l'image obtenue en supprimant ces 100 sillons.

FIGURE 2 – Détails d'implémentation du redimensionnement d'image.

2 Directives d'implémentation

On vous demande d'implémenter l'algorithme de redimensionnement décrit à la section précédente en respectant l'interface donnée dans `slimming.h`.

Pour vous aider, nous vous fournissons les fichiers suivants :

`mainSlimming.c` qui sert d'interface utilisateur. Il s'utilise comme suit

```
./slimming <inputImage> <outputPath> <nbPix>
```

où

- `inputImage` est le chemin d'accès vers une image au format `PNM`.
- `outputPath` est le chemin d'accès vers l'image résultante (format `PNM` également).
- `nbPix` est le nombre de pixels qu'il faut enlever à chaque ligne de l'image (le paramètre k dans la section précédente).

`PNM.h/c` une librairie permettant de lire, écrire et manipuler les images au format `PNM`.

`*.pnm` des images au format `PNM` pour réaliser vos tests.

Vous pouvez également convertir des images au format `PNM` grâce à des logiciels de retouche d'image ou à la librairie ImageMagick (`https://imagemagick.org/index.php`) en ligne de commande.

3 Rapport

1. Montrez qu'une approche exhaustive pour déterminer le sillon optimal serait de complexité exponentielle par rapport à la hauteur de l'image.
2. Proposez une approche gloutonne et montrez à l'aide d'un contre exemple qu'elle ne peut pas être optimale.
3. Donnez une formulation mathématique récursive de la fonction $C(i, j)$ ($1 \leq i \leq n$, $1 \leq j \leq m$). Précisez bien le ou les cas de base.
4. Représentez le graphe des appels récursifs pour un problème de petite taille.
5. Donnez le pseudo-code d'un algorithme *efficace* permettant de calculer le coût du sillon optimal.
6. Adaptez votre pseudo-code pour renvoyer l'image résultant d'une réduction de largeur de k pixels.
7. Analysez la complexité en temps et en espace de votre solution en fonction n et m, les dimensions de l'image, et k, la réduction de largeur.

4 Deadline et soumission

Le projet est à réaliser par groupes de **deux** étudiants pour le **08 mai 2020** à **23h59** au plus tard. **Indiquez les noms et matricules de chacun des membres sur le rapport et dans chacun des fichiers sources**.

Le projet est à remettre via la plateforme de Montéfiore :

`http://submit.montefiore.ulg.ac.be/`

Il doit être rendu sous la forme d'une archive `tar.gz` contenant :

1. Votre rapport au format PDF. Soyez bref mais précis et respectez bien la numération des (sous-)questions.
2. Le fichier `slimming.c`.

Respectez bien les extensions de fichiers ainsi que leur nom pour les fichier `*.c` (en ce compris la casse). Seule la dernière archive soumise sera prise en compte.

Vos fichiers seront évalués avec les flags habituels (`--std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes -DNDEBUG`)

Ceci implique que :
— Le projet doit être réalisé dans le standard C99.
— La présence de *warnings* impactera négativement la cote finale.
— Un projet qui ne compile pas avec cette commande sur ces machines recevra une cote nulle (pour la partie code du projet).

Un projet non rendu à temps recevra automatiquement une cote nulle. En cas de plagiat avéré, l'étudiant se verra affecter une cote nulle à l'ensemble du projet. Soyez bref mais précis dans votre rapport et respectez bien la numérotation des sous-questions de l'énoncé.

Les critères de correction sont précisés sur la page web du cours.

Bon travail !