Réalisation d'un panorama

Ludovic Lamarche
Quentin Perales
Elie Poussou

E.I.S.T.I
PAU

19 janvier 2014

Table des matières

Introduction

Chapitre 1

Les six fonctions du livrable2

1.1 Grayscale

La fonction Grayscale permet de transformer une image couleur en teintes de gris.
Dans une image en couleur, chaque pixel est codé par 3 composantes (Rouge, Vert et Bleu).
En grayscale, chaque pixel sera codé par une seule composante.
La formule utilisée est la suivante :
$pixelGrayscale = 0,299 * pixelRouge + 0,587 * pixelvert + 0,114pixelbleu$
On parcourt donc l'image couleur, et on applique cette formule à chaque pixel, c'est à dire à ces trois composantes.
Cette fonction effectue donc une moyenne pondérée des pixels rouges, verts et bleus.
L'image obtenue sera donc 3 fois plus petite, et en noir et blanc.

1.2 Binnary

Chaque pixel d'une image binaire ne peut avoir pour valeurs que 0 ou 1. La fonction binary nécessite un seuil et une image en teintes de gris, le seuil étant choisit par l'utilisateur.

On parcourt l'image en teintes de gris, et pour chaque pixel, si son instensité est supérieure au seuil, on lui alloue la valeur 0 qui correspond à la couleur blanche. Si l'intensité du pixel est supérieure au seuil, on lui alloue la valeur 1 qui correspond à la couleur noire.
Voici l'algorithme utilisé :

```
1          POUR TOUT les pixels de l'image
             SI le pixel >seuil
3            pixel=1
           FINSI
5        FINPOUR
```

1.3 Histogramme

L'histogramme d'une image permet d'illustrer la répartition des teintes d'une image.
Le but est de compter pour chaque intensité le nombre de pixels qui sont de cette intensité.

D'abord on crée un tableau de taille égale à la teinte maximale de l'image dont on veut obtenir l'histogramme.
Ensuite, on parcourt l'image et pour chaque pixel, on rajoute 1 dans la case du tableau qui correspond à l'instensité de ce pixel.

On peut résumer tout ceci grâce à l'algorithme suivant :

```
1 tab : tableau d'entiers de taille égale à la teinte maximale de
    l'image
```

```
POUR chaque pixel de l'image
  tab[teinte_pixel]++
FINPOUR
```

On peut choisir de tracer cet histogramme mais le tableau suffit.

1.4 Dilate

Pour la dilatation nous avons utilisé un algorithme qui utilise des "objects pixel" tiré du livre Machine Vision[2]. Un objet pixel est un groupe de neuf pixels formant un carré de taille 3x3 et qui respecte certaines conditions. Pour simplifier l'algorithme on nomme chaque case de 0 à 8.

3	2	1
4	8	0
5	6	7

Si on utilise les conventions de logique combinatoire, c'est à dire que le signe . signifie ET et le signe + signifie OU, les conditions d'un objet pixel se résume comme ceci :

$$\begin{aligned} &8.[((1+2+3).5+6+7).\overline{4}.\overline{0}] \\ &+[(1+0+7).(3+4+5).\overline{2}.\overline{6}] \\ &+[3.(5+6+7+0+1).\overline{2}.\overline{4}] \\ &+[1.(3+4+5+6+7).\overline{2}.\overline{0}] \\ &+[7.(1+2+3+4+5).\overline{0}.\overline{6}] \\ &+[5.(7+^{\circ}+1+2+3).\overline{4}.\overline{6}] \end{aligned}$$

Finallement, il suffit d'appliquer l'algorithme suivant :

```
POUR TOUT les pixels de l'image
  SI le pixel est un objet pixel ALORS
    copier le groupe de pixels dans l'image de destination
  FINSI
```

```
5 FINPOUR
```

Étant donné que cette méthode enlève beaucoup de pixels, il n'est pas nécessaire de faire une érosion en suivant. Cependant nous avons utilisé un autre algorithme moins restrictif pour l'erosion.

1.5 Erode

Pour l'érosion nous utilisons un masque. Nous testons un groupe de pixel et nous assignons la valeur 1 si tout les pixels dans le masque sont à 1. Sinon on retourne un 0. Nous avons pris directement l'algorithme d'opération pixel[1] :

```
1 destination : matrice de l'image érodée
  source : matrice de l'image à éroder
3 x,y,k1,k2 : variables d'incrémentation

5 POUR CHAQUE pixel de l'image FAIRE //x et y étant la position
    POUR k1 allant de -1 à 1 FAIRE
7     POUR k2 allant de -1 à 1 FAIRE
  destination[x][y] = destination[x][y] ET source[x+k1][y+k2]
9     FINPOUR
    FINPOUR
11 FIN POUR
```

1.6 Convolution [3]

La convolution permet d'appliquer un filtre de convolution à une image en échelle de gris. Le but est d'affecter pour chaque pixel de l'image des coefficients aux pixels autour de celui-ci.

La taille du filtre est variable, cependant, elle est toujours paire. Les filtres les plus utilisés sont les filtres de taille 3x3, mais parfois, les filtres de tailles

supérieures sont requis. Les filtres sont récupérés en début de fonction, une erreur est générée si le filtre n'est pas correct, c'est à dire si le fichier texte comporte des caractères autres que des chiffres ou nombres ou s'il n'y a pas assez de coefficients.

Afin d'expliquer l'algorithme qui mène à la convolution, nous allons utiliser un filtre 3x3 sur une image. Prenons le filtre F suivant :

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

F est appliqué à un pixel p de coordonnées (x, y). La valeur du pixel après la convolution sera alors

$$nouvelleValeur = 0 * p(x-1, y-1) + 1 * p(x, y-1) + 0 * p(x+1, y-1)$$
$$+ 1 * p(x-1, y) + 1 * p(x, y) + 1 * p(x+1, y)$$
$$+ 0 * p(x-1, y+1) + 1 * p(x, y+1) + 0 * p(x+1, y+1)$$

De plus, on crée un décalage pour éviter d'assigner des coefficients à des pixels hors image. Pour un filtre 3x3, le décalage est de 1 ; pour 5x5, il est de 3 ; pour 7x7, il est de 5. On peut donc dire que

$$decalage = \frac{taille - 1}{2} \tag{1.1}$$

Voici un exemple de convolution que l'on peut obtenir :

$$\begin{matrix} 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 48 & 52 & 56 & 58 & 60 \\ 56 & 60 & 65 & 70 & 75 \end{matrix} \quad \text{X} \quad \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad = \quad \begin{matrix} 35 & 40 & 41 & 45 & 50 \\ 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 56 & 60 & 65 & 70 & 75 \end{matrix}$$

Le but va être maintenant d'utiliser ces fonctions ainsi que d'autres algorithmes pour automatiser la création d'un panorama.

Chapitre 2

Notre algorithme d'automatisation d'un panorama

Après avoir vu les principales fonctions utiles à notre panorama, nous devons maintenant développer des méthodes pour pouvoir coder des algorithmes capables d'assembler des images les unes aux autres et ainsi retrouver l'image initiale.

2.1 La détection des points d'intérêts d'une image

2.1.1 Récupération des points clé

2.1.2 Algorithme de comparaison des points clé

2.1.3 fonction de collage des images

2.1.4 Notre algorithme principal

2.1.5 Comment améliorer cet algorithme ?

Conclusion

Bibliographie

[1] OPERATIONPIXEL@FREE.FR. *Traitement d'images*. 2010. URL : http://operationpixel.free.fr/pointinteret_fast.php.

[2] David VERSNON. *Machine Vision*. 1991. Chap. 4. URL : http://homepages.inf.ed.ac.uk/rbf/BOOKS/VERNON/Chap004.pdf.

[3] WIKIPÉDIA. *Kernel (Image processing)*. URL : http://en.wikipedia.org/wiki/Kernel_(image_processing).