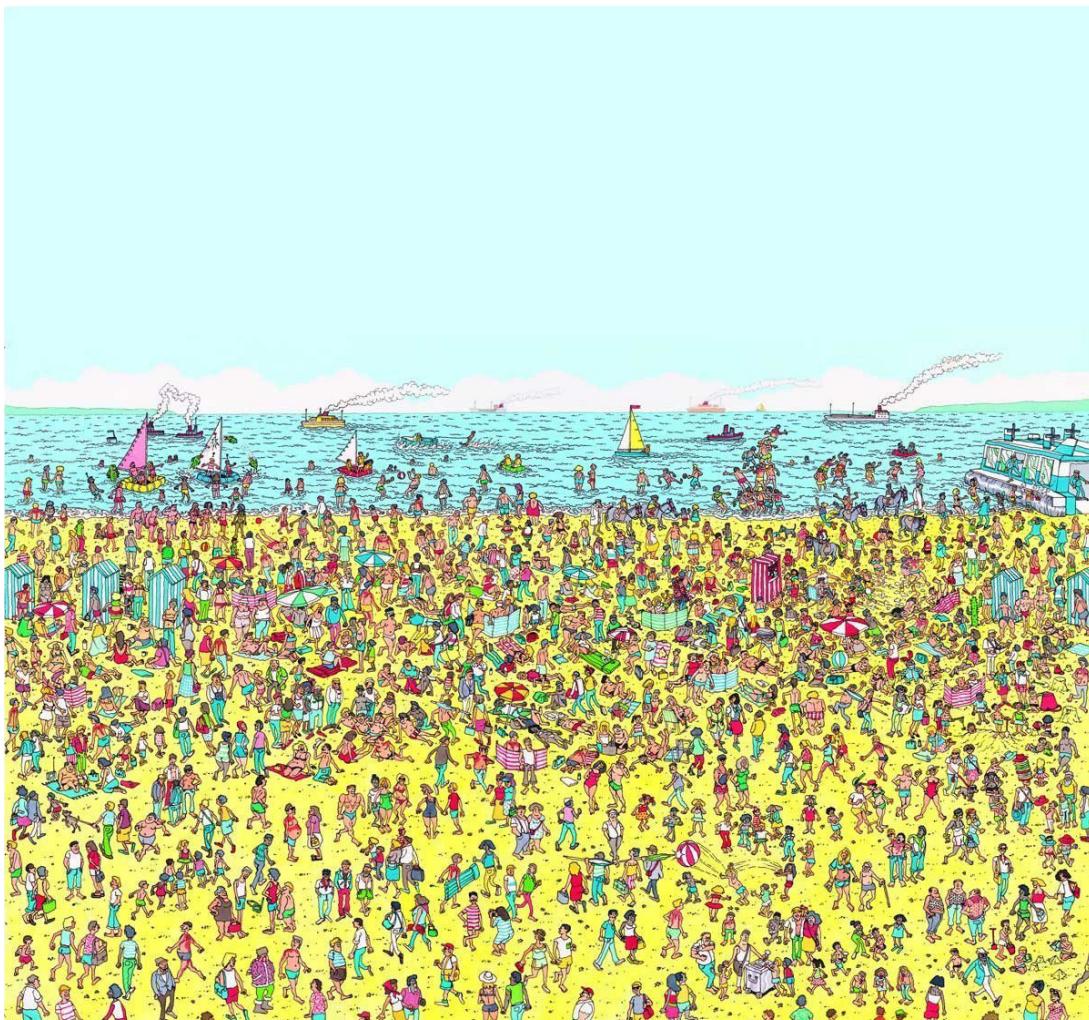


Algorithmique des images

TD n° 4

Détection de motifs



Le but de ce travail est de développer un programme permettant de rechercher un motif dans une image afin de trouver Charlie.

Pour ce TP nous utiliserons les structures `rgb_t`, `pgm_t` et `ppm_t` définies dans les TP précédents.

On suppose que l'image de départ I est de taille $M \times N$ et que le motif que l'on cherche K est de taille $m \times n$ ($m \leq M$ et $n \leq N$). Pour $0 \leq i < M$ et $0 \leq j < N$ on note $I(i, j)$ le pixel d'indice (i, j) de l'image I , ainsi l'image I est formée de l'ensemble des pixels $I = (I(i, j))_{\substack{0 \leq i < M \\ 0 \leq j < N}}$.

Exercice 1. Travail préliminaire

Q- 1.1 Question 1

Créer une fonction `pgm_t *empty_image(int value_max, int height, int width)` qui crée une image de type `pgm_t` de taille `height × width` dont la valeur des pixels peut être au maximum égale à `value_max`. On mettra la valeur de tous les pixels à 0.

Q- 1.2 Question 2

- ▶ Créer les fonctions `void highlight_rectangle_pgm(pgm_t *image, int x, int y, int height, int width)` et `void highlight_rectangle_ppm(ppm_t *image, int x, int y, int height, int width)` qui mettent en évidence un rectangle de taille `height × width` dont le coin supérieur gauche est situé sur le pixel d'indice (x, y) . La fonction assombrira tous les pixels qui ne sont pas dans le rectangle en divisant leur valeur par trois. La valeur des pixels à l'intérieur du rectangle restera inchangée. Cette fonction permettra de mettre en évidence Charlie lorsqu'on l'aura trouvé :



Exercice 2. Calcul de la Corrélation Croisée Normalisée

La première étape lors d'une recherche de motif est de mesurer la similitude entre le motif K et toutes les sous images de I de taille $m \times n$.

On rappelle que la sous-image de taille $m \times n$ d'indice (x, y) de I est l'image définie par $I_{x,y} = (I(x+i, y+j))_{\substack{0 \leq i < m \\ 0 \leq j < n}}$ ($i + m < M$ et $j + n < N$).

- ▶ Créer les fonctions `double average_pixels_pgm(pgm_t *image)` et `double_rgb average_pixels_ppm(ppm_t *image)` qui permettent de calculer la valeur moyenne des pixels de l'image `pgm` ou `ppm` passée en paramètre.

Remarque : dans le cas des images `ppm_t`, le résultat sera stocké dans un élément de type `double_rgb_t` défini ci-après :

```

1 struct double_rgb
2 {
3     double R, G, B;
4 };
5 typedef struct double_rgb double_rgb_t;
```

On rappelle que la moyenne des valeurs des pixels d'une image est calculée par la formule :

$$M(I) = \frac{1}{MN} \sum_{i,j} I(i,j)$$

```

1 /* Informations de debug */
2 average_pixels_ppm(puzzle1/charlie.ppm) = (128.80, 99.85, 93.62)
3 average_pixels_ppm(puzzle2/charlie.ppm) = (169.72, 144.49, 122.01)
```

- ▶ Créer les fonctions `double std_dev_pgm(picture *image, double average)` et `double_rgb std_dev_ppm(picture *image, double_rgb average)` qui permettent de calculer la fonction `std_dev`, définie ci-dessous, de l'image `pgm` ou `ppm` passée en paramètre.

$$\text{std_dev}(I) = \sqrt{\sum_{i,j} (I(i,j) - M(I))^2}$$

Remarque : on suppose que la moyenne des valeurs des pixels de l'image est donnée en paramètre par le paramètre `average`.

```

1 /* Informations de debug */
2 std_dev_ppm(puzzle1/charlie.ppm) = (1676.34, 1657.63, 1320.54)
3 std_dev_ppm(puzzle2/charlie.ppm) = (2437.72, 2549.94, 2435.18)
```

- ▶ Créer les fonctions, dont la signature est donnée ci-dessous, et qui permettent de calculer la CCN de la sous-image d'indice (x, y) de `image` et du motif `pattern`. La fonction prendra également en paramètre la moyenne des valeurs des pixels du motif `average_pattern` et leur déviation standard `std_dev_pattern`.

- `unsigned char NCC_pgm(pgm_t *image, pgm_t *pattern, int x, int y, double average_pattern, double std_dev_pattern)`
- `unsigned char NCC_ppm(ppm_t *image, ppm_t *pattern, int x, int y, double_rgb average_pattern, double_rgb std_dev_pattern)`

Remarque :

- la CCN est définie comme un nombre dans l'intervalle $[-1, 1]$. Afin de la stocker sur un `unsigned char` on renormalisera le résultat de sorte à ce soit un entier dans l'intervalle $[0, 255]$.
- on calculera la CCN de deux images `ppm` en calculant la norme des CCN de chacune des composantes R, G et B :

$$\text{CCN}(I, K) = \frac{\sqrt{\text{CCN}_n(I, K).R^2 + \text{CCN}(I, K).G^2 + \text{CCN}(I, K).B^2}}{\sqrt{3}}$$

```

1  /* Informations de debug */
2  x = 231, y = 76;
3  NCC_ppm(puzzle1/puzzle1.ppm, puzzle1/charlie.ppm, x, y) = 105
4  NCC_ppm(puzzle2/puzzle2.ppm, puzzle2/charlie.ppm, x, y) = 123

```

- ▶ Créer la fonction `pgm_t *compute_NCC(pgm_t *image, pgm_t *pattern)` qui renvoie l'image `pgm` de taille $(M - m) \times (N - n)$ dont le pixel d'indice (i, j) contient la valeur de la NCC entre `pattern` et la sous image d'indice (i, j) de `image`.

Exercice 3. Seuillage et suppression des non-maximaux

- ▶ Créer une fonction `pgm_t *thresholding(pgm_t *image, int threshold)` qui effectue le seuillage de l'image `pgm` `image` par la valeur `threshold` en calculant l'image `res` de même taille que `image` telle que :

$$\text{res}(i, j) = \begin{cases} 255 & \text{si } \text{image}(i, j) \geq \text{threshold} \\ 0 & \text{sinon} \end{cases}$$

- ▶ Créer une fonction `pgm_t *local_maxima(picture *image)` qui prend en paramètre une image `pgm` et qui renvoie une nouvelle image `res` de même type et de même taille que `image` et telle que :

$$\text{res}(i, j) = \begin{cases} 0 & \text{s'il existe un couple } (k, l) \in \{-1, 0, 1\}^2 \text{ tel que : } \text{image}(i + k, j + l) > \text{image}(i, j) \\ 255 & \text{sinon} \end{cases}$$

- ▶ Créer une fonction `pgm_t *pattern_matching(pgm_t *image, pgm_t *pattern)` qui renvoie une copie de l'image `pgm` `image` dans laquelle le motif `pattern` est mis en évidence à l'aide de la fonction `highlight_rectangle`.
- ▶ Créer un programme prenant en paramètre l'image dans laquelle chercher Charlie et l'image de Charlie à chercher et qui crée une copie de l'image dans laquelle Charlie est mis en évidence.
- ▶ Modifier votre programme afin de pouvoir chercher plusieurs motifs en même temps de sorte à pouvoir également faire apparaître Blanchebarbe et Pouah en plus de Charlie.

