TP3 – Perception des formes dans une image

Afin que vous n'ayez qu'un seul fichier à rendre pour ce TP, au lieu de créer un fichier pour chaque fonction que vous aurez à écrire, un fichier nommé `fonctions_TP3_proba` vous est fourni pour que vous complétiez les différentes fonctions qui y sont présentes.

Introduction

Dans une image binaire de synthèse, on considère les pixels comme des variables aléatoires iid. Un pixel reçoit la valeur 0 (pixel noir) avec la probabilité p ou la valeur 1 (pixel blanc) avec la probabilité $1-p$ (*loi de Bernouilli* de paramètre p). Malgré l'indépendance entre pixels, des agglomérats de pixels noirs, en forme de carrés, peuvent apparaître par le fait du hasard. La probabilité $\mathcal{P}(p,t)$ pour qu'un carré de côté t soit noir est égale à p^{t^2}. Elle diminue très rapidement si les carrés deviennent plus grands, c'est-à-dire lorsque t croît, ou si les pixels noirs se raréfient, c'est-à-dire lorsque p décroît. Or, un carré noir se distingue d'autant mieux qu'il est plus grand ou que l'image est plus claire (cf. figure 1). Cet exemple très simple illustre le principe général de la perception des formes dans une image : une forme est d'autant plus perceptible qu'elle est moins probable.

FIGURE 1 – Ces images binaires de taille 64×64, tirées selon une loi de Bernouilli de paramètre p, contiennent un carré noir unique de côté $t = 4$ ou 5. La probabilité d'apparition $\mathcal{P}(p,t) = p^{t^2}$ du carré décroît de gauche à droite, ce qui le rend d'autant plus perceptible : notre œil est attiré par les événements les moins probables !

Lancez le script `carres_noirs`, qui simule une séquence d'images binaires aléatoires de taille 64×64 tirées selon une loi de Bernouilli de paramètre p, jusqu'à ce qu'une de ces images contienne au moins un carré noir de côté t. Ce script teste plusieurs valeurs de t, pour $p = 0,5$ fixé. Attention : la recherche est de plus en plus longue, puisque la probabilité d'apparition $\mathcal{P}(p,t) = p^{t^2}$ d'un carré noir décroît fortement lorsque t croît. Néanmoins, plus l'attente est longue, plus la forme « saute aux yeux » dans l'image affichée.

Comme cela a déjà été vu dans le TP2, les variables aléatoires que constituent les pixels d'une image *naturelle* ne sont pas iid, mais sont au contraire fortement corrélées. C'est la raison pour laquelle les formes perceptibles sont beaucoup plus courantes dans une image naturelle que dans une image aléatoire. En revanche, il s'agit en général de formes approchées. Il est donc nécessaire de donner une définition plus souple à la notion de forme. Dans une image binaire aléatoire de taille $T \times T$ tirée selon une loi de Bernouilli de paramètre p, la probabilité pour qu'un carré de côté t, c'est-à-dire constitué de $N = t^2$ pixels, contienne *exactement* $n \leqslant N$ pixels noirs, est donnée par la *loi binomiale* $\mathcal{B}(p,N,n) = \mathrm{C}_N^n\, p^n (1-p)^{N-n}$, où C_N^n désigne le nombre de combinaisons de n éléments parmi N. Par conséquent, la probabilité pour qu'un tel carré contienne *au moins* n pixels noirs s'écrit :

$$\sum_{k=n}^{N} \mathcal{B}(p,N,k) = \sum_{k=n}^{N} \mathrm{C}_N^k\, p^k (1-p)^{N-k} = 1 - \sum_{k=0}^{n-1} \mathrm{C}_N^k\, p^k (1-p)^{N-k} \tag{1}$$

La dernière somme de (1) peut se calculer à l'aide de la fonction `binocdf` de Matlab (`doc binocdf`).

Exercice 1 : détection de pixels voisins ayant des gradients parallèles

Détecter les « alignements » dans une image consiste à faire une esquisse constituée de segments de droites. De tels segments se situent là où le module $\|\nabla I\|$ du gradient du niveau de gris I est élevé. Pour détecter les alignements, il faut donc commencer par calculer le gradient du niveau de gris ∇I, par exemple avec la fonction `gradient` (axe des abscisses orienté vers la droite, axe des ordonnées vers le bas, unité de longueur = pixel).
Une fois ce gradient calculé, seuls les pixels où sa norme est supérieure à un seuil sont sélectionnés. Parmi ces pixels, on cherche à constituer des ensembles E (cf. figure 2-gauche) tels que :

- chaque ensemble E soit connexe, au sens des « 8 plus proches voisins » ;
- le gradient $G_{i,j} = \nabla I(i,j)$ en un pixel (i,j) de E soit parallèle à la somme G_Σ des gradients calculée sur les pixels de l'ensemble E en cours de construction, c'est-à-dire $\frac{G_{i,j}}{\|G_{i,j}\|} \cdot \frac{G_\Sigma}{\|G_\Sigma\|} \geqslant \cos\alpha$, où $\alpha > 0$ désigne l'angle maximal autorisé entre les vecteurs $G_{i,j}$ et G_Σ.

Après avoir lu attentivement le script `exercice_1`, complétez la fonction `ensemble_E_recursif`, qui permet de construire, par appels récursifs, les ensembles E à partir de « germes ». Les figures 2-centre et 2-droite montrent le résultat de l'exécution de ce script sur l'image `Piree.png` (les ensembles E sont affichés sous différentes couleurs).

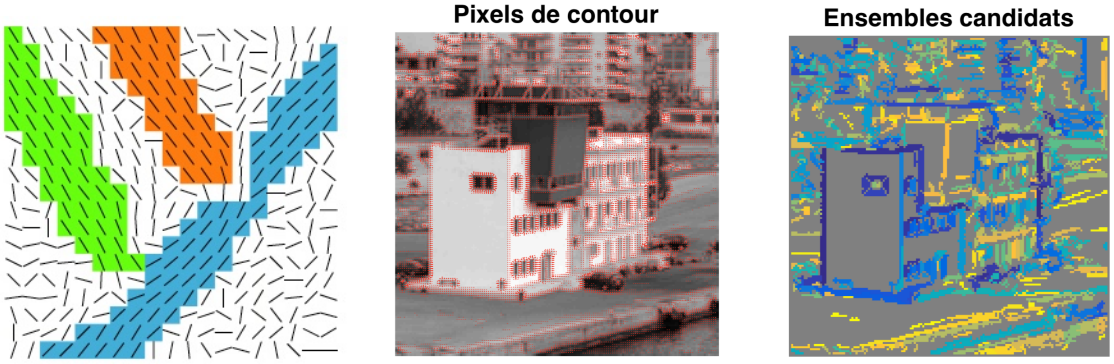

FIGURE 2 – À gauche : exemple comportant trois ensembles candidats E. Au centre : image sur laquelle le gradient du niveau de gris est superposé aux pixels de contour. À droite : ensembles candidats E associés.

Exercice 2 : détection des alignements dans une image

Si la variable aléatoire « gradient du niveau de gris » en un pixel suivait une loi uniforme, la probabilité p pour qu'une réalisation fasse un angle inférieur à α, relativement à une direction de référence, vaudrait $p = \frac{2\,\alpha}{2\,\pi} = \frac{\alpha}{\pi}$.
Soit E un des ensembles de pixels constitués à l'étape précédente. On définit R comme le plus petit rectangle englobant de E, de côtés parallèles aux axes principaux de sa *matrice d'inertie* (cf. cours d'Analyse de Données au 2nd semestre). Cette matrice est une généralisation de la matrice de variance/covariance, pour laquelle les coordonnées (x_i, y_i) sont pondérées par $\pi_i = \|\nabla I\|_i$:

$$\bar{x} = \frac{1}{\Pi}\sum_{i=1}^{n} \pi_i x_i \;;\; \bar{y} = \frac{1}{\Pi}\sum_{i=1}^{n} \pi_i\, y_i \;;\; M_{1,1} = \frac{1}{\Pi}\sum_{i=1}^{n} \pi_i\,(x_i-\bar{x})^2 \;;\; M_{1,2} = \frac{1}{\Pi}\sum_{i=1}^{n} \pi_i\,(x_i-\bar{x})(y_i-\bar{y}) \;;\; \text{etc.}\ldots \quad (2)$$

où $\Pi = \sum_{i=1}^{n} \pi_i$. Écrivez la fonction `matrice_inertie`, qui doit retourner les coordonnées $(\bar{x}, \bar{y})$ du centre d'inertie C de E ainsi que sa matrice d'inertie M, puis lancez le script `test_matrice_inertie` en guise de vérification.
Remarque : Il faut bien penser à permuter les colonnes de E (x est orienté verticalement et y horizontalement), à l'aide de la fonction `fliplr` si besoin, pour coïncider avec l'ordre des axes du gradient (i est orienté horizontalement et j verticalement).

Une fois le rectangle R déterminé en passant dans le repère des axes principaux de E, on compte le nombre n de pixels (i, j) de E où $\nabla I(i, j)$ fait un angle inférieur à α avec la direction du petit axe de R. La probabilité pour que cela se produise en n pixels du rectangle R *au moins* est donnée par (1), où $p = \frac{\alpha}{\pi}$ et $N = \text{card}(R)$. Le critère pour déterminer si un ensemble candidat E forme un alignement détectable s'écrit alors ($\epsilon \ll 1$ est un seuil) :

$$1 - \sum_{k=0}^{n-1} \mathrm{C}_N^k \, p^k (1-p)^{N-k} < \epsilon \tag{3}$$

Écrivez la fonction `calcul_proba`, qui doit calculer le premier membre de (3) (le nombre de pixels N doit être entier !). Exécutez ensuite le script `exercice_2` afin de visualiser l'esquisse de l'image avec les candidats E conservés, et voir le nombre de candidats rejetés.

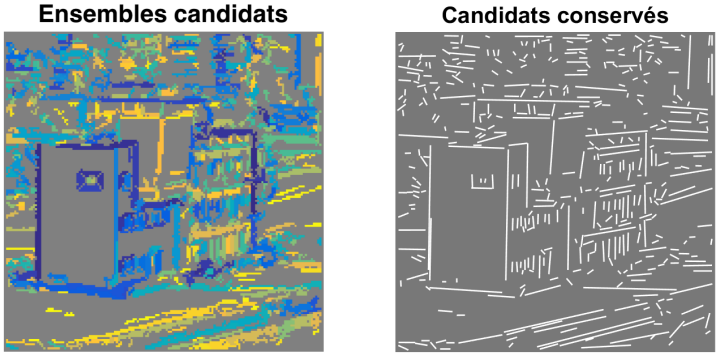

FIGURE 3 – À gauche : ensembles candidats E associés. À droite : ensembles candidats E conservés.

Une fois mis au point sur l'image `Piree.png`, vous pourrez relancer dans l'ordre les scripts `exercice_1` et `exercice_2` en sélectionnant les images `chaises.png` et `Morlaix.png`, ou en important n'importe quelle autre image de votre choix.