TP5 – Détection de la ligne d'horizon dans une image

Le script `exercice_1.m` affiche l'image d'un bateau dont le pont est recouvert de panneaux solaires qui constituent un pavage régulier du pont. À partir de cette seule image, peut-on savoir si le pont était horizontal au moment de la prise de vue ? Une réponse erronée consisterait à vérifier si l'horizon est parallèle aux lignes de l'image : cela dépend uniquement de la façon dont la photographie a été prise, et n'a rien à voir avec l'horizontalité du pont. Le script `exercice_1.m` affiche également l'image (binaire) des contours obtenue avec la fonction `edge` de Matlab. Les bords des panneaux solaires forment des segments de droites qui semblent concourir en deux points particuliers appelés « points de fuite ». La droite passant par les points de fuite, que l'on appelle la « ligne de fuite », donne une indication sur l'orientation du pont (vous étudierez cela en 3A, dans le cours de *Vision 3D*). Il faut donc localiser les points de fuite, en commençant par détecter les droites passant par le plus grand nombre de pixels de contour. Une méthode très classique de reconnaissance de formes permettant d'effectuer cette tâche, qui date de 1962, est la **transformation de Hough**.

Exercice 1 : transformation de Hough

Une droite du plan Δ peut être caractérisée par un couple de paramètres (ρ, θ) définis comme suit :

- θ est l'angle polaire du vecteur $\vec{v}$ orthogonal à Δ, de norme 1, tel que $\theta \in [0, \pi[$ (cf. figure 1).
- À partir d'un point $P = (x, y)$ quelconque de Δ, on peut calculer le second paramètre $\rho = x \cos\theta + y \sin\theta$. Il est facile de donner une interprétation géométrique de ρ, qui représente la « distance à l'origine » : $|\rho| = OQ$, où Q désigne la projection orthogonale de O sur Δ, et ρ a le signe du produit scalaire $\vec{v} \cdot \overrightarrow{OQ}$.

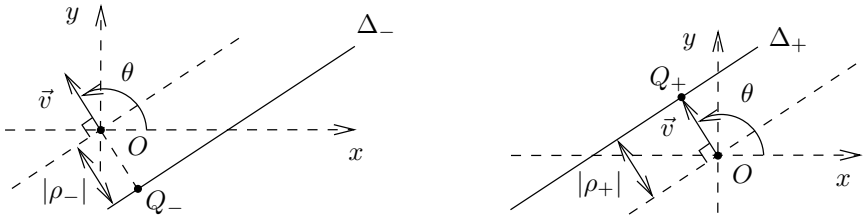

FIGURE 1 – Les droites Δ_- et Δ_+ correspondent aux paramètres $(\rho_-, \theta) = (-1, 130)$ et $(\rho_+, \theta) = (1, 130)$.

On choisit comme origine O du repère le centre de l'image, et comme unité de longueur la distance entre pixels voisins. L'axe des abscisses est orienté vers la droite, l'axe des ordonnées vers le bas (orientations par défaut en Matlab). Les droites du plan qui intersectent l'image sont paramétrées par des valeurs bornées de ρ et θ : $\rho \in [-d, d]$, où d désigne la demi-diagonale de l'image ; $\theta \in [0, \pi[$. On discrétise ces deux intervalles :

- On arrondit ρ à l'entier le plus proche, à l'aide de la fonction `round` de Matlab. Ce paramètre peut donc prendre N_ρ valeurs correspondant aux valeurs entières situées dans l'intervalle $[-d, d]$.
- On limite θ aux $N_\theta = 180$ angles de l'intervalle $[0, \pi[$ correspondant à des valeurs entières en degrés, à savoir $\theta = k\,\pi/180$, $k \in [0, 180[$.

Parmi les $N_\theta N_\rho$ droites du plan qui intersectent l'image, on cherche celles qui passent par le plus grand nombre de pixels de contour. Pour ce faire, on introduit une matrice C à N_ρ lignes et N_θ colonnes, dont l'élément $C(i, j)$ comptabilise le nombre de points de contour situés sur la droite de paramètres $(\rho(i), \theta(j))$.

Complétez le script `exercice_1.m` de façon à appliquer l'algorithme suivant : pour chaque pixel de contour de coordonnées (x, y), et pour chaque angle $\theta \in [0, \pi[$, calculer $\rho = \text{round}(x \cos\theta + y \sin\theta)$ et incrémenter l'élément de la matrice C correspondant au couple (ρ, θ) (on peut dire que le pixel (x, y) a « voté » pour la droite définie par ces paramètres). En guise de vérification, affichez la matrice C sous la forme d'une pseudo-image.

Exercice 2 : recherche des maximums locaux de la matrice C

La détection des n droites passant par le plus grand nombre de pixels de contour se fait en recherchant les n maxima locaux les plus élevés de la matrice C, par exemple en parcourant en boucle la séquence suivante :

1. Rechercher l'élément $C(i_{\max}, j_{\max})$ ayant recueilli le plus de suffrages.
2. Stocker le couple $(\rho(i_{\max}), \theta(j_{\max}))$ sur une ligne d'une matrice M de taille $n \times 2$.
3. Mettre toutes les cases de C à 0, sur une fenêtre de taille $(2\,T+1) \times (2\,T+1)$ centrée en $(i_{\max}, j_{\max})$.

Complétez le script `exercice_2.m` de façon à ce que les n maxima locaux les plus élevés soient indiqués sur la pseudo-image C par des croix rouges, et les droites correspondantes soient superposées à l'image originale.

Exercice 3 : localisation des points de fuite et de la ligne de fuite

Les droites détectées forment deux faisceaux, dont les centres constituent deux points de fuite. Il est nécessaire de partitionner en deux les croix rouges détectées à l'exercice 2. Or, pour un point F quelconque du plan, de coordonnées polaires (ρ_F, θ_F) et de coordonnées cartésiennes (x_F, y_F), il est facile de caractériser les droites passant par F, c'est-à-dire le faisceau de droites de centre F, car les paramètres (ρ, θ) de ces droites vérifient :

$$\rho = x_F \cos\theta + y_F \sin\theta \tag{1}$$

c'est-à-dire, comme $x_F = \rho_F \cos\theta_F$ et $y_F = \rho_F \sin\theta_F$:

$$\rho = \rho_F \cos(\theta - \theta_F) \tag{2}$$

Effectivement, deux lignes claires de forme sinusoïdale sont visibles sur la pseudo-image C. Comme les croix rouges correspondant à un même faisceau occupent une petite portion de sinusoïde, elles semblent alignées.

Complétez le script de nom `exercice_3.m` de façon à scinder les croix rouges en deux classes. Vous devez imaginer une méthode permettant de former deux sous-ensembles de croix rouges quasi-alignées. Affichez les deux classes par des croix de couleurs différentes, et faites de même pour les droites affichées sur l'image originale.

Ensuite, le script `exercice_3.m` estime la position de chaque point de fuite (intersection approchée de droites quasi-concourantes), puis trace la ligne de fuite, c'est-à-dire la droite passant par les deux points de fuite. Si la ligne de fuite coïncide avec l'horizon, alors le pont était effectivement horizontal au moment de la prise de vue. Attention : contrairement à une idée très répandue, même si la Terre était plate, il y aurait un horizon (qui serait légèrement plus haut).

Exercice 4 : vérification de la ligne d'horizon d'un gyroscope (facultatif)

Lancez le script `exercice_4.m`, qui affiche l'image d'une scène d'intérieur, à laquelle est superposée une ligne bleue censé correspondre à la ligne d'horizon. Cette image et cette ligne d'horizon proviennent de la base de données *Toulouse Vanishing Points Dataset* développée dans l'équipe VORTEX de l'IRIT :

`http://ubee.enseeiht.fr/dokuwiki/doku.php?id=public:toulousevpdataset`

Chaque image de la base de données est prise par un capteur comportant un gyroscope. Un fichier Matlab d'extension `.mat`, de nom identique à celui de l'image, contient les données complémentaires fournies par le gyroscope. La ligne d'horizon fournie par le gyroscope est-elle suffisamment précise ?

En vous inspirant des exercices précédents, complétez le script `exercice_4.m` de manière à afficher la ligne d'horizon obtenue à partir des points de fuite. Conclusion : le gyroscope était-il bien réglé ?