MI204 - Reconnaissance d'Images
Corrigé du Contrôle 2021

1. CONVOLUTION : On considère les quatre noyaux de convolution suivants, où l'origine (i.e. le coefficient attaché au pixel courant) est l'élément apparaissant en *italique* dans la matrice :

$$A = \begin{bmatrix} 1 & -2 & 1 \\ 2 & -\mathit{4} & 2 \\ 1 & -2 & 1 \end{bmatrix}, B = \begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -6 & \mathit{0} & 6 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathit{0} \end{bmatrix}, D = \begin{bmatrix} 1 \\ -4 \\ \mathit{6} \\ -4 \\ 1 \end{bmatrix}.$$

Pour chacun de ces noyaux de convolution, décrivez le plus précisément possible l'effet qu'il produit sur une image, et éventuellement, quelle opération ou grandeur il est censé approximer.

Corrigé - 1 point par Noyau :

— **A :** C'est un noyau de somme nulle, correspondant à un filtre dérivateur. On peut le décomposer comme la convolution du noyau $\begin{bmatrix} 1 & -\mathit{2} & 1 \end{bmatrix}$, qui correspond à l'estimation de la dérivée seconde horizontale, par le noyau $\begin{bmatrix} 1 & \mathit{2} & 1 \end{bmatrix}^T$, qui correspond à un lissage vertical. La convolution de l'image I par le noyau A sert donc à approximer $\frac{\partial^2 I}{\partial x^2}$.

— **B :** Encore un noyau dérivateur de somme nulle, qui correspond à la convolution du filtre lisseur vertical $\begin{bmatrix} 1 & 4 & \mathit{6} & 4 & 1 \end{bmatrix}^T$ par le noyau $\begin{bmatrix} -1 & \mathit{0} & 1 \end{bmatrix}$, qui correspond à l'estimation de la dérivée première horizontale. La convolution de l'image I par le noyau B sert donc à approximer $\frac{\partial I}{\partial x}$.

— **C :** La convolution par le noyau C calcule la somme de 9 pixels dans le voisinage du pixel translaté de vecteur $(-3,-3)$ par rapport à l'origine. Cela se traduit donc par : (1) une amplification du niveau de gris d'un facteur 9, (2) un floutage de l'image, et (3) une translation de l'image de 3 pixels en diagonal.

— **D :** Encore un noyau dérivateur de somme nulle. La symétrie du noyau indique qu'il s'agit d'une dérivée d'ordre pair. Et en effet D est égal à la convolution du noyau $\begin{bmatrix} 1 & -\mathit{2} & 1 \end{bmatrix}^T$ (dérivée seconde verticale) par lui-même. La convolution de l'image I par le noyau D sert donc à approximer $\frac{\partial^4 I}{\partial y^4}$.

2. INVARIANCE : Que désigne la notion d'invariance par rotation pour un détecteur ou un descripteur calculé sur une image ? Comment l'obtient-on ? Donnez quelques exemples. Quelles sont les limites pour des images de scènes 3d ?

Corrigé - 1 point par question :

— L'invariance par rotation *pour un détecteur* désigne la capacité du détecteur à sélectionner le même point dans plusieurs images (répétabilité), quelque soit la rotation 2d qui affecte l'apparence de ce point dans les différentes images. L'invariance par rotation *pour un descripteur* désigne le fait que le vecteur descripteur est identique lorsqu'il est calculé avant ou après une rotation de l'image.

— L'invariance par rotation est obtenue, soit en utilisant des grandeurs intrinsèquement invariantes dans le calcul, soit en calculant une orientation propre attachée au point, qui servira de référence.

— Par exemple les grandeurs telles que niveau de gris, norme du gradient, laplacien ou courbure de l'isophote sont intrinsèquement invariantes par rotation et peuvent être utilisées

pour détecter des points clefs, ou dans un descripteur local. Un autre exemple (descripteur SIFT) consiste à calculer la distribution des orientations autour d'un point-clef en considérant l'orientation dominante du point-clef (blob) comme origine des angles.
— La limitation principale vient du fait qu'une rotation 3d d'un objet de la scène ne se traduit pas en général par une rotation 2d une fois projetée dans l'image. L'invariance par rotation ne garantit donc pas à elle seule la robustesse aux transformations rigides qui peuvent affecter l'objet.

3. ACP ET BAYES : Expliquez comment on peut combiner l'Analyse en Composantes Principales et la classification bayésienne en général. Quel intérêt cela peut-il présenter ? Imaginez un exemple qui vous semble pertinent en reconnaissance d'images.

   Corrigé :

   — Pour combiner ACP et classification bayésienne, il suffit de remplacer les données d'observations $\{X_i\}_i$ exprimées dans leur base d'origine par les vecteurs réduits $\{X_i^r\}_i$, exprimés dans la base de l'ACP. **(1 point)**
   — L'intérêt est multiple ; cela induit d'abord une réduction significative du coût de calcul de l'apprentissage et de l'inférence bayésienne, l'ACP réalisant en général une réduction de dimension des vecteurs de données d'un ou plusieurs ordres de grandeur. Ensuite, on peut s'attendre à une meilleure discrimination des différentes classes, dans la mesure où les (premières) composantes des vecteurs après ACP maximisent la variance des données et sont donc plus significatives des différences entre les données (Attention toutefois au fait que ces différences ne sont pas forcément corrélées à la classe). **(1,5 point)**
   — Question ouverte, on peut imaginer par exemple la segmentation de textures, à partir de la réduction de petites imagettes (patches) de taille 17×17 (vecteurs de dimension 289), à une dizaine de composantes principales. On peut ensuite apprendre à classifier les pixels en fonction des composantes principales associées à leur voisinage 17×17 pour distinguer differents types de surface (route, forêt, bâtiment, ciel, etc.). On peut alors utiliser, soit un modèle gaussien, en calculant pour chaque classe un vecteur moyen et une matrice de covariance, soit un classifieur de Bayes naïf, en calculant pour chaque classe les histogrammes composante par composante. **(1,5 point)**

4. HOUGH ET HORIZON : La détection de la ligne d'horizon est un problème intéressant en vision par ordinateur. Sachant que l'horizon se caractérise en général par la droite horizontale vers laquelle convergent toutes les lignes de fuite induites par la perspective (voir exemple Figure 1), proposez une méthode fondée sur le gradient et la transformée de Hough pour détecter automatiquement la position de la ligne d'horizon dans l'image.

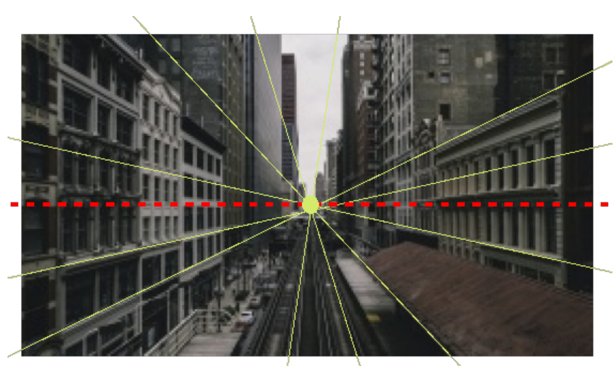

FIGURE 1 – Lignes de fuite et horizon en perspective 1-point.

   Corrigé - 4 points :

   Il y a plusieurs solutions possibles. Il faut d'abord noter qu'on ne peut pas détecter l'horizon directement comme une droite horizontale, car il ne forme pas une droite dans l'image en général !

La solution la plus simple est sans doute de faire une transformée en 2 temps : d'abord une transformée "1-to-1" en calculant en chaque point P le gradient ∇I . On en déduit l'équation polaire de la droite (si elle existe, c'est-à-dire si $||\nabla I||$ est assez grand) qui passe par P, avec $\theta = \arg \nabla I$ l'orientation de la normale, et $\rho = \frac{|P.\nabla I|}{||\nabla I||}$ la distance à l'origine. Puis, en prenant les N plus grand maxima de la transformée $\{\theta_n, \rho_n\}_n$, on fait une transformée de Hough "inverse" "1-to-many", en votant pour les droites correspondantes dans l'espace image. En faisant l'hypothèse que la majorité des droites sont des lignes de fuite de la perspective, les points qui ont reçu le plus de votes doivent être des points de fuite, alignés sur l'horizon.

Mais la solution la plus efficace - car directe - est sans doute d'utiliser une transformée de Hough "2-to-1", en tirant au hasard un grand nombre de couples de points $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ dans l'image I, et en calculant les paramètres de la droite (si elle existe) qui passe par chacun d'eux, soient (θ_1, ρ_1) et (θ_2, ρ_2). Si $\theta_1 \neq \theta_2$, alors les deux droites s'intersectent en $M = (x_M, y_M)$, avec $y_M = \frac{\rho_2 \cos\theta_1 - \rho_1 \cos\theta_2}{\sin(\theta_2 - \theta_1)}$. Si l'on suppose que l'horizon est horizontal, il suffit donc de voter dans l'espace des paramètres associé à $y \in [0, h]$, où h est la hauteur de l'image.

5. Réseau de Neurones : Quelle est la particularité d'un réseau de neurones convolutif (CNN) par rapport à un réseau de neurones en général ? Quels sont ses avantages en général ? Quel intérêt particulier présente-t-il pour l'analyse des images ? Quelles contraintes implique-t-il par rapport à la structure des données d'entrées ?

Corrigé - 1 point par question :

— Dans un réseau de neurones convolutif, *un même neurone* s'applique de façon répétée et invariante par translation sur toute la longueur, la surface, ou le volume de la donnée d'entrée.

— L'avantage est de *limiter le nombre* de neurones et de rendre l'architecture du réseau *indépendante* de la taille des données d'entrée.

— Ces réseaux ont particulièrement du sens pour les images, l'invariance par translation étant un principe de base en analyse d'images, et parce que les caractéristiques de bas niveau fournies par la géométrie locale (niveau moyen, contraste, orientation, courbure...) peuvent être convenablement estimées par des convolutions, ce qui est corroboré par des arguments biologiques (traitements rétiniens) ou statistiques (ACP des images naturelles).

— Les CNN peuvent s'appliquer à d'autres données d'entrées que des images, mais on ne peut les justifier que s'il existe une *métrique régulière* sur ces données justifiant de limiter la combinaison à des données "proches" spatialement (ou temporellement).