3

Extraction de caractéristiques dans les images

3.1 Introduction

Ce chapitre concerne l'extractions des caractéristiques dans l'image. Les trois types de caractéristiques que nous allons étudier sont les contours, les lignes et les points. Les pixels contours sont des pixels pour lesquels l'intensité de l'image change brusquement, et les contours (ou les segments contours) sont des ensembles de pixels contours connectés. La détection de contours est une méthode de traitement d'image locale conçue pour détecter les pixels de contours. Une ligne peut être considérée comme un segment contours dans lequel l'intensité du fond de chaque côté de la ligne est beaucoup plus élevée ou bien inférieure à l'intensité des pixels de la ligne. De même, un point peut être considéré comme une ligne dont la longueur et la largeur sont égales à un pixel.

3.2 Extraction de contours

La détection des changements d'intensité afin de détecter des contours peut être obtenue en utilisant des dérivées du premier ou second ordre. Dans cette section, nous introduisons les dérivées du premier ordre. Les dérivées de second ordre seront abordées ultérieurement dans ce chapitre.

3.2.1 Le gradient d'image et ses propriétés

L'outil choisi pour trouver l'amplitude et la direction d'un contour à la position (x, y) d'une image, f, est le *gradient*, noté ∇f, et défini par le vecteur

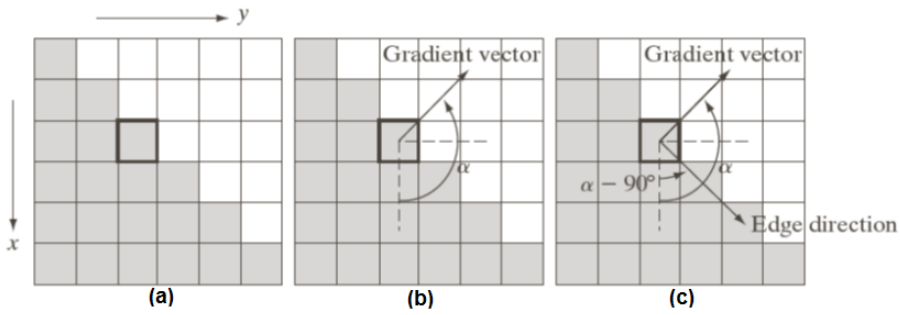

FIG. 3.1: Utilisation du gradient pour déterminer l'amplitude et la direction du contour en un point. Chaque carré de la figure représente un pixel.

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{3.1}$$

La propriété principale qui caractérise le vecteur gradient est qu'il se positionne vers la direction du changement maximale de la fonction f au point (x, y).

L'amplitude (longueur) du vecteur ∇f, notée $M(x, y)$, est donnée par:

$$M(x, y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{3.2}$$

La valeur $M(x, y)$ représente le taux de changement dans la direction du vecteur gradient.

Notez que g_x, g_y et $M(x, y)$ sont des images de la même taille que l'image originale. Ces images sont créées en variant les coordonnées (x, y) sur toutes les positions possibles des pixels de l'image f. L'image résultante est appelée *image du gradient*, ou simplement *gradient*. Les opérations somme, carré et racine carrée, dans l'Eq. (3.12), sont des *opérations vectorielles*.

La direction du vecteur gradient est donnée par l'angle:

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right) \tag{3.3}$$

mesuré par rapport à l'axe des abscisses. Noter que $\alpha(x, y)$ est aussi une image de la même taille que l'image originale. La Fig. 3.1 montre une zone zommée d'une image contenant un segment d'un contour. Noter que la direction du contour est perpendiculaire à la direction du vecteur gradient au point où le gradient est calculé.

3.2.2 Opérateurs du gradient

Pour obtenir le gradient d'une image, il est nécessaire de calculer les dérivées partielles $\frac{\partial f}{\partial x}$ et $\frac{\partial f}{\partial y}$ à chaque position de l'image. Par conséquent, il doit y avoir une approximation numérique des dérivées partielles sur un voisinage d'un point.

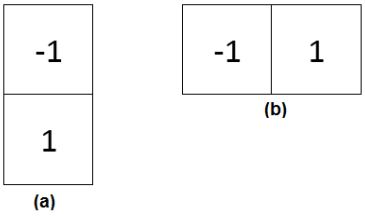

FIG. 3.2: Masques unidimensionnels utilisés pour implémenter les équations (3.4) et (3.5).

On a

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{3.4}$$

et

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y-1) - f(x,y) \tag{3.5}$$

Ces deux équations peuvent être implémentées pour toutes les valeurs possibles de x et y en filtrant $f(x,y)$ avec les masques unidimensionnels de la Fig. 3.2.

Les gradients *Roberts* (1965) sont l'une des premières tentatives de l'utilisation des masques 2-D avec une prise en compte des contours diagonales. Considérons la région 3×3 de la Fig. 3.3(a). Les opérateurs Roberts sont basés sur l'implémentation des diagonales:

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5) \tag{3.6}$$

et

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6) \tag{3.7}$$

Ces dérivées peuvent être implémentées en filtrant une image avec les masques des Figs. 3.3 (b) et (c).

Les masques de taille 2×2 sont simples, mais ils ne sont pas aussi intéressants pour calculer la direction des contours que les masques symétriques sur le point central. Ces masques prennent en considération la nature des données sur les côtés symétriques du point central et portent ainsi plus d'informations sur la direction d'un contour. Les approximations numériques les plus simples des dérivées partielles utilisant des masques de taille 3×3 sont données par

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{3.8}$$

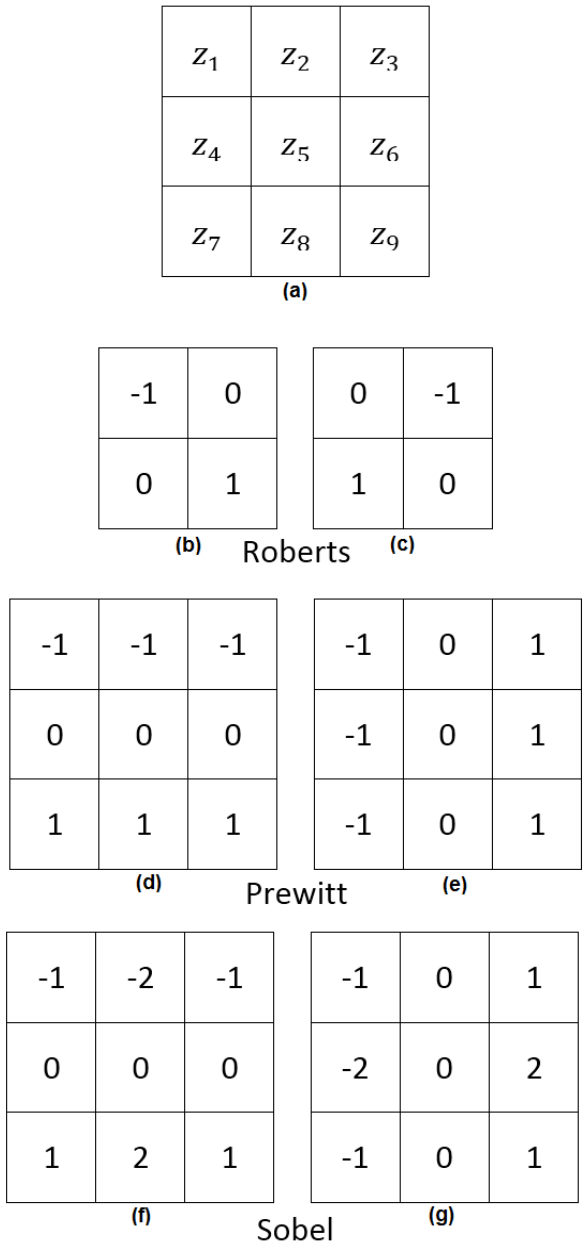

FIG. 3.3: Région d'image de taille 3×3 (les variables z sont des valeurs d'intensité) et plusieurs types des masques utilisés pour calculer le gradient au point z_5.

et

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{3.9}$$

Dans ces formules, la différence entre la 3ième et la 1ière ligne de la zone 3×3 se rapproche de la dérivée par rapport à x et la différence entre la 3ième et la 1ière colonne se rapproche

de la dérivée par rapport y. Intuitivement, ces approximations sont plus précises que celles obtenues avec les opérateurs Roberts. Les Eqs. (3.8) et (3.9) peuvent être implémentées en filtrant l'image f avec les masques des Figs. 3.3(d) et (e). Ces masques s'appellent *les opérateurs Prewitt* (1970).

Une petite variation des deux équations précédentes utilise un poids de 2 pour le coefficient central, est donnée par:

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{3.10}$$

et

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{3.11}$$

Il peut être démontré que l'utilisation du coefficient 2 avec le pixel central fournit un lissage de l'image. Les Figs. 3.3(f) et (g) représentent les masques utilisés pour implémenter les Eqs. (3.10) et (3.11). Ces masques s'appellent *les opérateurs Sobel* (1970).

Les masques Prewitt sont plus simples en termes d'implémentation par rapport aux masques Sobel, mais la légère différence de calcul entre eux n'est généralement pas un problème. Le fait que les masques de Sobel possèdent de meilleures caractéristiques de réduction du bruit (lissage) les rend préférables car la suppression du bruit est un problème à traiter lors de l'utilisation des dérivées. Notez que la somme des coefficients de tous les masques de la Fig. 3.3 est égale à zéro, ce qui donnera une réponse de zéro dans avec les régions où l'intensité est constante, conclusion attendu d'un opérateur de dérivée.

Les masques que nous venons de présenter sont utilisés pour obtenir les composantes g_x et g_y du gradient à chaque pixel de l'image. Ces deux dérivées partielles sont ensuite utilisées pour estimer l'amplitude et la direction du contour. Le calcul de l'amplitude du gradient nécessite que g_x et g_y se combinent de la manière indiquée dans l'Eq. (3.12). Cependant, cette implémentation n'est pas optimale grâce à la charge de consommation requise par les carrés et les racines carrées. Une solution utilisée fréquemment consiste à rapprocher l'amplitude du gradient par des valeurs absolues de la manière suivante:

$$M(x, y) \approx |g_x| + |g_y| \tag{3.12}$$

Il est possible de modifier les masques 3×3 de la Fig. 3.3 afin d'avoir les réponses les plus fortes selon les directions diagonales. La Fig. 3.4 montre les deux autres masques Prewitt et Sobel nécessaires pour détecter les contours dans les directions diagonales.

3.2.3 Méthode (algorithme) de Canny

Le détecteur de contour de Canny (1986) est l'un des algorithmes de détection de contours parmi les plus utilisés. Cet algorithme possède trois caractéristiques fondamentales:

1) Il est robuste au bruit. Pour y arriver, il préfiltre l'image à l'aide d'un filtre gaussien.

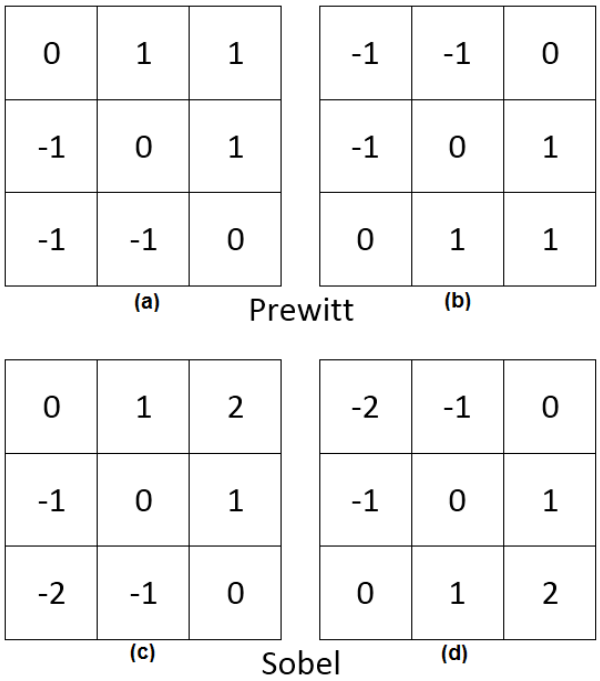

FIG. 3.4: Masques Prewitt et Sobel pour la détection des contours diagonales.

2) Détecte des contours filiformes. Pour y arriver, il supprime les non-maximums.

3) Élimine les edge pixels isolés et les edge segments trop petits. Pour ce faire, il applique un seuillage par hystérésis.

L'algorithme de Canny pour la détection de contour se compose des étapes suivantes:

3.2.3.1 Réduction du bruit

La première étape consiste à réduire le bruit de l'image originale. Ceci permet d'éliminer les pixels isolés qui pourraient induire de fortes réponses lors du calcul du gradient, conduisant ainsi à de faux positifs. Un filtrage gaussien 2-D est utilisé avec l'opérateur de convolution:

$$G(x,y) = \exp\left(\frac{x^2+y^2}{2\sigma^2}\right) \tag{3.13}$$

et un exemple de masque 5×5 discret avec $\sigma = 1.4$:

$$h = \frac{2}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3.14}$$

3.2.3.2 Calcul du gradient

L'étape suivante est d'appliquer un gradient qui retourne l'intensité des contours. L'opérateur utilisé permet de calculer le gradient suivant les directions x et y, il est composé de deux masques de convolution, un de dimension 3×1 et l'autre 1×3 :

$$g_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$g_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{3.16}$$

La valeur du gradient en un point:

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{3.17}$$

Cette valeur est approximée par la formule :

$$M(x, y) = |g_x| + |g_y| \tag{3.18}$$

3.2.3.3 Direction des contours

Les orientations des contours sont déterminées par la formule :

$$\alpha(x, y) = \arctan(\frac{g_y}{g_x}) \tag{3.19}$$

Nous obtenons finalement une carte des gradients en chaque point de l'image accompagnée des directions des contours.

3.2.3.4 Suppression des non-maxima

La carte des gradients obtenue précédemment fournit une intensité à chaque point de l'image. Une forte intensité indique une forte probabilité de présence d'un contour. Toutefois, cette intensité ne suffit pas à décider si un point correspond à un contour ou non. Seuls les points correspondant à des *maxima locaux* sont considérés comme contours, et sont conservés pour la prochaine étape de la détection. Un maximum local est présent sur les extrema du gradient, c-à-d là où sa dérivée s'annule.

3.2.3.5 Seuillage des contours

La différenciation des contours sur la carte générée se fait par *seuillage à hystérésis*. Cela nécessite deux seuils, un seuil haut et un seuil bas; qui seront comparés à l'intensité du gradient à chaque pixel. Le critère de décision est le suivant. Pour chaque point, si l'intensité de son gradient est:

- Inférieur au seuil bas, le point est rejeté;
- Supérieur au seuil haut, le point est accepté comme formant un contour;
- Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté.

Une fois ceci est réalisé, l'image obtenue est binaire avec d'un côté les pixels appartenant aux contours et de l'autre côté les pixel non contours.

3.3 Extraction de lignes

Dans cette section, nous présentons une approche basée sur une hypothèse qui suppose que les ensembles de pixels se situent sur des courbes d'une forme spécifique. Une fois détectées, ces courbes forment les contours ou les limites des régions d'intérêt.

Etant donnés n points dans une image, supposons que nous voulons trouver les sous-ensembles de ces points qui se situent sur des lignes droites. Une solution possible est de trouver d'abord toutes les lignes déterminées par chaque paire de points, puis de trouver tous les sous-ensembles de points proches de lignes particulières. Cette approche consiste à trouver $n(n-1)/2 \sim n^2$ lignes, puis à effectuer $(n)(n(n-1))/2 \sim n^3$ comparaisons de chaque point à toutes les lignes. Cette approche est la plus couteuse en termes de temps de calcul mais elle est la plus triviale.

Hough (1962) a proposé une approche alternative appelée *la transformation de Hough*. Considérons un point (x_i, y_i) dans le plan xy et l'équation générale d'une droite de la forme, $y_i = ax_i, +b$. Une infinité de droites traversent (x_i, y_i), mais elles satisfont toutes l'équation $y_i = ax_i + b$ en variant les valeurs de paramètres a et b. Cependant, en réécrivant cette équation comme $b = -x_i a + y_i$ et considérant le plan ab (appelé aussi *espace de paramètre*) cela va donner l'équation d'une *seule droite* pour un point fixe (x_i, y_i). En plus, un deuxième point (x_j, y_j) a également une droite dans l'espace de paramètre associé à celui-ci et, à moins qu'ils ne soient parallèles, cette droite coupe la droite associée à (x_i, y_i) à un certain point (a', b'), où a' est la pente et b' l'intercept de la droite contenant les deux points (x_i, y_i) et (x_j, y_j) dans le plan xy. En fait, tous les points sur cette droite ont des droites dans l'espace des paramètres qui se croisent à (a', b'). La Fig. 3.5 illustre ces concepts.

En principe, les droites de l'espace des paramètres ab correspondantes à tous les points (x_k, y_k) dans le plan xy peuvent être tracées, et les droites principales dans ce plan peuvent être trouvées en identifiant les points dans l'espace de paramètres où un grand nombre de droites de l'espace de paramètre s'intersectent. Cependant, une difficulté pratique avec cette approche, c'est que a (la pente d'une droite) approche l'infini lorsque la droite approche la

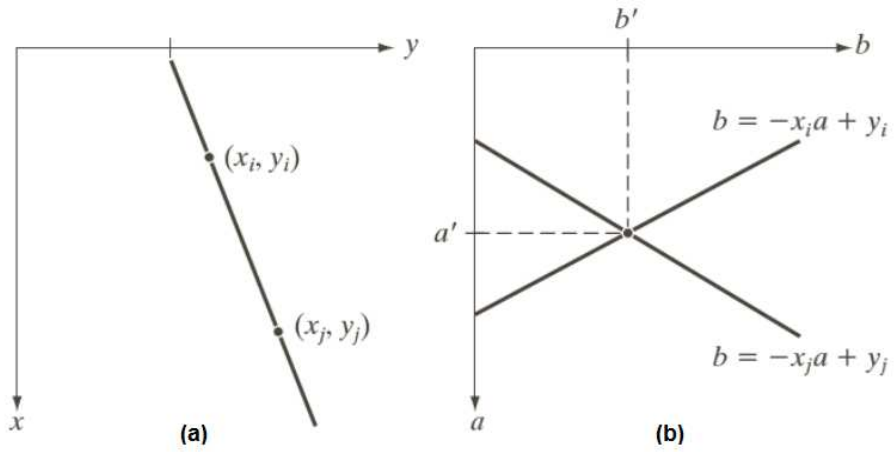

FIG. 3.5: Illustration du concept de l'space de paramètres. (a) Plan xy. (b) Espace de paramètres.

direction verticale. Une façon de contourner cette difficulté est d'utiliser la représentation normale d'une droite:

$$x\cos\theta + y\sin\theta = \rho \tag{3.20}$$

La Fig. 3.6(a) illustre l'interprétation géométrique des paramètres ρ et θ. Une droite horizontale a $\theta = 0°$, avec ρ égal à l'intercept x positive. De même, une ligne verticale a $\theta = 90°$, avec ρ est égal à l'intercept y positive, ou $\theta = -90°$, avec ρ égal à l'intercept y négative. Chaque courbe sinusoïdale de la Fig. 3.6(b) représente la famille des droites qui traversent un point particulier (x_k, y_k) dans le plan xy. Le point d'intersection (ρ', θ') de la Fig. 3.6(b) correspond à la droite qui traverse les deux points (x_i, y_i) et (x_j, y_j) de la Fig. 3.6(a).

L'efficacité de calcul de la transformation de Hough provient du fait de partitionner l'espace des paramètres $\theta\rho$ en *cellules accumulatrices*, tel que montré dans la Fig. 3.6(c), où (ρ_{min}, ρ_{max}) et $(\theta_{min}, \theta_{max})$ sont les intervalles des valeurs des paramètres: $-90° \leq \theta \leq 90°$ et $-D \leq \rho \leq D$, où D représente la distance maximale entre les coins d'une diagonale dans une image. La cellule aux coordonnées (i, j), avec la valeur d'accumulateur $A(i, j)$, correspond au carré associé aux coordonnées espace-paramètre (ρ_i, θ_j). Initialement, ces cellules sont mises à zéro. Ensuite, pour chaque point non arrière-plan (x_k, y_k) dans le plan xy, on met θ égale à chacune des valeurs de subdivision sur l'axe des θ et calculer le ρ correspondant en utilisant l'équation $\rho = x_k cos\theta + y_k sin\theta$. Les valeurs de ρ résultantes sont ensuite arrondies sur la valeur de la cellule autorisée la plus proche le long de l'axe ρ. Si un choix de θ_p résulte en une solution ρ_q, alors on aura $A(p, q) = A(p, q) + 1$. À la fin de cette procédure, une valeur de P dans $A(i, j)$ signifie que les P points dans le plan xy se situent sur la ligne $\rho_i = x\cos\theta_j + y\sin\theta_j$. Le nombre de subdivisions dans le plan $\rho\theta$ détermine la précision de la colinéarité de ces points. Il peut être montré que le nombre de calculs dans la méthode qui vient d'être discutée est linéaire en termes de n, le nombre de points non arrière-plan dans le plan xy.

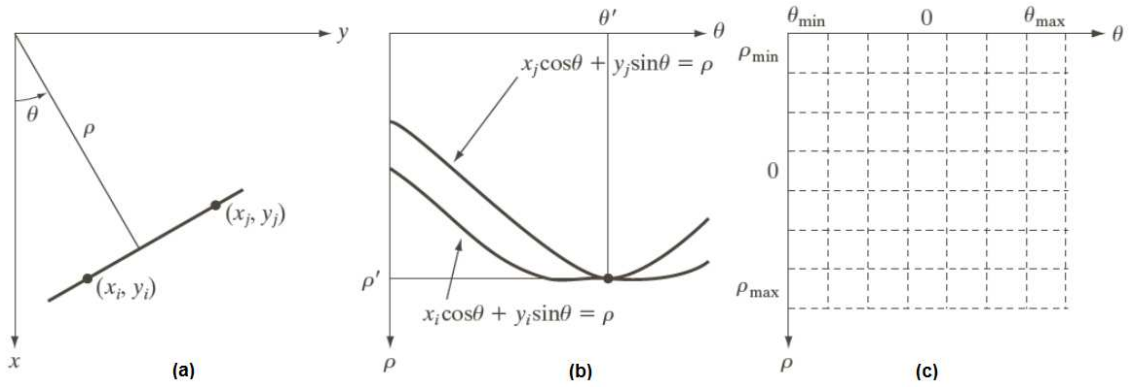

FIG. 3.6: (a) Les paramétres (ρ, θ) d'une droite dans le plan xy. (b) Courbes sinusoïdales dans le plan $\rho\theta$; le point d'intersection (ρ', θ') correspond à la droite passant par les points (x_i, y_i) et (x_j, y_i) dans le plan xy. (c) Division du plan $\rho\theta$ en cellules accumulatrices.

3.4 Extraction de coins

La détection de coins est une approche utilisée dans les systèmes de vision par ordinateur pour extraire certains types de descripteurs et déduire le contenu d'une image. La détection de coins est fréquemment utilisée dans la détection de mouvement, le suivi dans la vidéo, la mosaïque d'image, la modélisation 3D et la reconnaissance d'objets. La détection de coins se chevauche avec le sujet de la détection des points d'intérêt.

Harris considère le différentiel du score de coin le par rapport à la direction. Ce score de coin est souvent appelé autocorrélation. Cependant, les mathématiques dans l'article indiquent clairement que la somme des différences au carré est utilisée.

Sans perte de généralité, nous supposerons qu'une image bidimensionnelle en niveaux de gris est utilisée. Soit cette image donnée par I. Considérez prendre un correctif d'image sur la zone (u, v) et le déplacer par (x, y). La somme pondérée des différences carrées (SSD) entre ces deux patchs, notée S, est donnée par:

$$S(x, y) = \sum_u \sum_v w(x, y)(I(u + x, v + y) - I(u, v))^2 \tag{3.21}$$

$I(u + x, v + y)$ peut être approché par le développement de Taylor. Soit I_x et I_y, les dérivées partielles de I, tel que

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y \tag{3.22}$$

ceci donne l'approximation:

$$S(x, y) \approx \sum_u \sum_v w(x, y)(I_x(u, v)x + I_y(u, v)y)^2, \tag{3.23}$$

qui peut être écrite sous la forme matricielle:

$$S(x,y) \approx (x\ y)\mathbf{A}\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.24}$$

Où $\mathbf{A}$ est appelé un *tenseur de structure*,

$$\mathbf{A} = \sum_u \sum_v w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{3.25}$$

Cette matrice est une *matrice de Harris*, et les parenthèses angulaires représentent la moyenne (c'est-à-dire la sommation (u,v)). Si une fenêtre circulaire $w(u,v)$ (ou une fenêtre circulaire pondérée, telle que *la fenêtre Gaussienne*) est utilisée, la réponse sera *isotrope*.

Un coin (ou en général un point d'intérêt) se caractérise par une grande variation de S dans toutes les directions du vecteur $(x\ \ y)$. En analysant les valeurs propres de $\mathbf{A}$, cette caractérisation peut s'exprimer de la manière suivante: $\mathbf{A}$ doit avoir deux "grandes" valeurs propres pour un point d'intérêt. Sur la base des grandeurs des valeurs propres, les décisions suivantes peuvent être prises en fonction de cet argument:

1) Si $\lambda_1 \approx 0$ et $\lambda_2 \approx 0$ alors ce pixel (x,y) n'a pas de caractéristiques de point d'intérêt.

2) Si $\lambda_1 \approx 0$ et λ_2 a une grande valeur positive, un contour est détecté.

3) Si λ_1 et λ_2 ont de grandes valeurs positives, un coin est détecté.

Harris note que le calcul exact des valeurs propres est coûteux en termes de temps de calcul, car il nécessite le calcul d'une racine carrée, et suggère plutôt la fonction suivante M_c, où κ est un paramètre de sensibilité à ajuster:

$$M_c = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(\mathbf{A}) - \kappa\ trace^2(\mathbf{A}) \tag{3.26}$$

Par conséquent, l'algorithme ne doit pas nécessairement calculer la décomposition en valeurs propres de la matrice $\mathbf{A}$. Il suffit d'évaluer le déterminant et la trace de la matrice $\mathbf{A}$ pour détecter les coins (ou les points d'intérêt en général).

3.5 Conclusion

Dans ce chapitre, plusieurs caractéristiques de l'image ont été introduites, en commençant par les contours, passant par les lignes et terminant par les points d'intérêts. Dans le chapitre suivant, nous allons aborder un thème très important pour le traitement d'image qui est la restauration et reconstruction d'image.

3.6 Travaux pratiques N° 3

3.6.1 Objectifs

- Implanter le filtrage d'image pour la réduction de bruit.
- Amélioration du contraste et détection de contours.

3.6.2 Enoncé

Dans ce projet, vous devez implémenter trois opérations d'amélioration de la qualité d'une image et une opération de détection de contours (les images de test vous seront fournies):

1) **Implanter le filtrage moyen, Gaussien et médian**

   Ici vous devez implanter une fonction de réduction de bruit 'bruit_reduction' qui aura trois paramètres d'entrée: l'image bruitée, la taille du filtre et le genre de filtre (ex. 0 pour le filtre moyen, 1 pour le filtre Gaussien et 2 pour le filtre médian). La fonction doit retourner l'image filtrée comme paramètre de sortie.

2) **Amélioration du contraste d'une image**

   Ici vous devez implanter l'opération d'amélioration de contraste en utilisant le Laplacien:

$$g(x,y) = f(x,y) + c\|\nabla^2 f\| \tag{3.27}$$

   Il faut implanter une fonction 'ameliorer_constraste' qui a en paramètre d'entrée: l'image à améliorer et le paramètre c, et en paramètre de sortie: l'image résultat.

3) **Détection de contours**

   Ici vous devez implanter une fonction 'contours_detection' qui prend comme paramètres d'entrée une image et une valeur de seuil, et en paramètre de sortie la carte des contours avec l'algorithme de Canny.

3.6.3 Astuces avec Matlab

Vous pouvez utiliser les fonctions **mean**, **median** et **fspecial** de MATLAB.