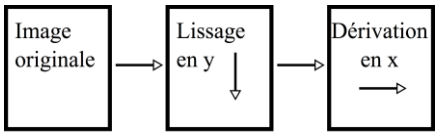
TD 6 UEE Vision

Dans ce TP, vous serez amenés à estimer des gradients, les directions à la fois des contours et du gradient puis les contours dans des images. Détecter les contours est important dans le domaine du traitement des images, notamment pour la restauration ou la détection et le suivi d'objets.

La détection de contours dans les images avec des filtres d'ordre 1 (dérivée première), on utilise une combinaison de filtre passe haut et bas. Pour la dérivée en *x*, il est nécessaire de lisser en *y* et de dériver en *x*, et inversement pour la dérivée en *y*.



Calcul de la dérivée de l'image en x

1) Surface image

Chargez l'image « bureau256.png » que l'on nommera *J*. La visualiser, repérer des contours. Créer une sous-image de *J*, appelée *S,* de taille 40x40 environ (important) puis afficher la surface image avec la commande :

>> figure, surf(double(S)) ; shading interp ;

Vous pouvez remarquer que les parties claires de *J* sont situées en hauteur tandis que les plus sombres vers le bas. Le contours de l'image sont situés là où la surface image possède des pentes, plus la pente est forte, plus le gradient sera marqué.

Affichez le ligne 95 de *J* sous forme de courbe (*>> help plot*). Quelles sont les valeurs prises par la courbe ? Créer une image couleur de *J*, appelée *Jc*, où l'on affichera la ligne 95 en **rouge**.

2) Calcul de gradient avec les masques de Prewitt

Calculer les dérivées d'image avec les masques : $Mx = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ et $My = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$
(pour la dérivée en *x* et en *y* respectivement) :

>> Jx = filter2(Mx, J) /6;
>> Jy = filter2(My, J) /6;

D'après-vous, pourquoi il est important de diviser par 6 ou par 3 le résultat ?

Affichez *Jx* et *Jy :*
>> figure(1);imagesc(Jx),colormap(gray), title('Image derivee en X');
Que remarquez-vous à propos des contours horizontaux sur *Jx* ?
Même question à propos des contours verticaux sur *Jy*.

Affichez le ligne 95 de *Jx* sous forme de courbe. En parallèle, afficher la ligne 95 en couleur sur *Jx*. Quelles sont les valeurs prises par la courbe ? Est-ce cohérent pour vous ?

3) Norme du gradient

A partir de *Jx* et *Jy*, calculer et visualiser la norme du gradient (calcul pixel par pixel) : $G = \sqrt{J_x^2 + J_y^2}$.
Visualiser cette image (pour les visualiser en sombre, on peut utiliser la fonction d'un des TDs précédents : *f_inverse*). Normalisez *G* puis seuillez *G* dans l'image *Gs* afin de faire ressortir les contours et affichez *Gs*.

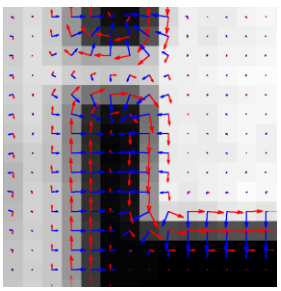
Trouver une technique pour afficher les contours seuillés (en blanc) sur l'image originale.
Maintenant, créer une image en couleur à partir de l'image *J* où l'on fera apparaître les contours en vert.

4) Direction du gradient

La direction du gradient est notée η et donnée par (calcul pixel par pixel):
>> eta = atan(Jy./Jx); % quand *Jx* est proche de 0, on rajoute une petite valeur par exemple «eps».
>> figure, imagesc(eta),colormap(gray), title('Gradient direction');

Noter que les directions des contours sont perpendiculaires à η, on peut tout visualiser sous forme de vecteurs en tapant les commandes suivantes **sur une sous image notée *S*** **(important)** :

→ sur l'image *S*, comme illustré :
>>figure, clf ; imagesc(S) ; colormap(gray) ; axis image; hold on;
>>quiver(-Sx, Sy);
>>quiver(Sy, -Sx,'r');



→ sur une sous image de l'image du gradient *G* :
>>figure,clf;imagesc(G);colormap(gray);axis image; hold on;
>>quiver(-Sx,Sy); quiver(Sy, Sx,'r');

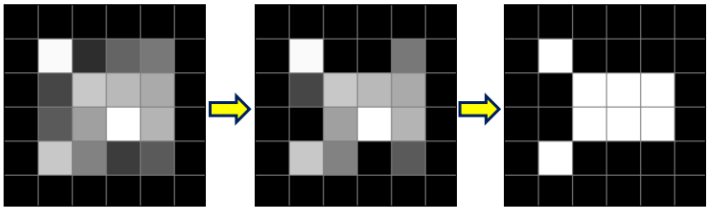
NW	N	NE
W	*P*	E
SW	S	SE

The current pixel value *P* is preserved only if it corresponds to the highest value along the 4 following axis:
- N-S
- NE-SW
- E-W
- SE-NW

(a) Non-maxima suppression technique in 4 directions inside a 3×3 mask

0	0	0	0	0	0
0	250	45	100	120	0
0	70	200	185	170	0
0	90	160	255	180	0
0	200	130	60	90	0
0	0	0	0	0	0

Pixel values of a processed image



Pixel intensities of a processed image | Non-maxima suppression using mask in (a) | Thresholded image: values > 150

(b) Different steps to obtain a thresholded image with non-maxima suppression in 4 directions

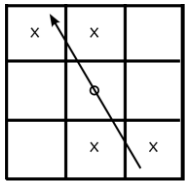
5) Extraction de maxima locaux selon les 4 directions

Créer la matrice à gauche en (b) dans l'image ci-dessus. Pour calculer simplement les maximas locaux dans un masque 3x3, il suffit de vérifier que chaque point est un maximum local selon les 4 directions N-S, NE-SW, E-W et SE-NW :

- S'il est maximum dans au moins une des 4 directions, le point est conservé
- Sinon le point prend la valeur 0.

Ensuite le résultat peut être seuillé pour obtenir une image binaire.
Reprendre la partie 3). Sur l'image *G*, pour chaque pixel : vérifier si la valeur du pixel est un maximum local selon les 4 directions N-S, NE-SW, E-W et SE-NW, dans ce cas garder la valeur du pixel.
Seuillez le résultat et affichez les contours en vert sur l'image originale.
Tester sur l'image « bureau256.png » puis « laine.png ».

6) Extraction de maxima locaux dans la direction du gradient



A partir des résultats des questions 3) et 4), utilisez les fonctions permettant d'extraire les maximas de *G* dans la direction du gradient. Les fonction à télécharger sont *directionalNMS* et *components2orientations*. Ainsi, pour chaque pixel de *G*, ces fonctions permettent de calculer si le pixel considéré est un maximum dans la direction η, comme illustré ci-contre.
Le résultat doit être des contours de largeur 1 pixel, en utilisant les fonctions suivantes (voir campus) :

```
>> [NMS] =  directionalNMS(Jx,Jy); % besoin de directionalNMS et components2orientations
>> Gradient_max= Ggray .* NMS;
```

Enfin, seuiller *Gradient_max* et afficher les contours en couleur sur l'image originale à l'aide d'une fonction.
Quelles différences faites-vous avec la partie 5) ?

7) Extraction de contours sur des images en couleur (calcul pixel par pixel)

Soient (R_x, V_x, B_x) les dérivées en *x* et (R_y, V_y, B_y) en *y* des canaux rouge, vert et bleu (R, V, B).
Di Zenzo [1] a proposé d'estimer le gradient sur les images couleurs à l'aide d'un tenseur multispectral **T :**

$$\mathbf{T} = \begin{pmatrix} (R_x^2 + V_x^2 + B_x^2) & (R_x R_y + V_x V_y + B_x B_y) \\ (R_x R_y + V_x V_y + B_x B_y) & (R_y^2 + V_y^2 + B_y^2) \end{pmatrix} \quad \text{On note : } \mathbf{T} = \begin{pmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_x\mathbf{I}_y & \mathbf{I}_y^2 \end{pmatrix}$$

Les valeurs propres du tenseur permettent de déterminer des contours pour chaque pixel de l'image [2], le gradient couleur est donnée par $\sqrt{\lambda_1}$, sachant : $\lambda_1 = \frac{1}{2}\left(\mathbf{I}_x^2 + \mathbf{I}_y^2 + \sqrt{\left(\mathbf{I}_x^2 - \mathbf{I}_y^2\right)^2 + \left(2 \cdot \mathbf{I}_x\mathbf{I}_y\right)^2}\right)$

Comparer les contours obtenus sur l'image «legoe.bmp» en couleurs avec ceux obtenus sur la même image en niveau de gris (*>> Ig = rgb2gray(I) ;*). Pour cela, afficher les deux résultats extraits des deux images sur l'image en couleur (→ créer une nouvelle fonction qui affiche des contours en vert sur une image couleur).
(Optionnel) Pour information, la direction du gradient couleur est $\eta = \frac{1}{2} arctan\left(2 * \left(I_x I_y\right)^2, I_{xx}^2 - I_{yy}^2\right)$.

8) Contours en utilisant la Gaussienne

On nomme par σ l' écart type de la gaussienne. On peut utiliser directement la fonction « fspecial » d'octave. Voici les différentes étapes à appliquer. i) Lisser l'image avec une gaussienne. ii) Calculer, visualiser *Ix* et *Iy*. iii) Calculer la norme du gradient ainsi que son angle η puis les maxima locaux dans la direction η, comme expliqué ci-dessus. iv) Calculer les non-maximas locaux et afficher en vert les contours sur l'image « bureau256.png ». v) Tester avec différentes valeurs de σ ainsi que sur l'image « laine.png ».

Enfin, calculer la norme du gradient d'une image couleur à l'aide de la gaussienne.