Opérations bas niveau sur les images

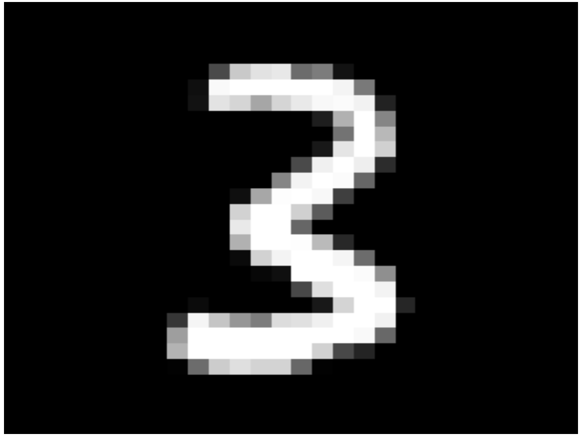
- La représentation sous forme de pixels a des désavantages
  - elle est lourde, c.-à-d. coûteuse en mémoire
    - 1024x1024 pixels de 8 bits (en niveau de gris) = 1 MB / image
    - 1024x1024 pixels de 24bits (canaux RGB) = 3 MB / image
  - elle contient plus d'information qu'on en a besoin
    - pour détecter une voiture dans une image, la couleur n'est pas utile
    - la scène (arrière plan) dans laquelle se trouve un objet à détecter peut être ignorée
- On aimerait appliquer des **opérations bas niveau simples** (**prétraitement**) sur les images, afin d'y **extraire l'information pertinente** pour la tâche à résoudre

Hugo Larochelle et Froduald Kabanza

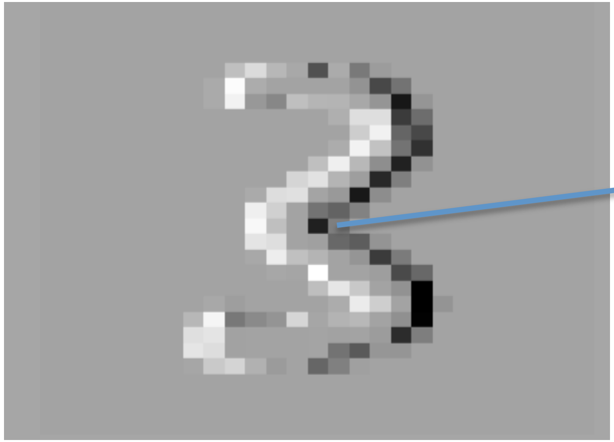
Gradient d'image

- Pour détecter si un pixel est sur la frontière d'un contour, on peut regarder la valeur relative des pixels autour de ce pixel

- Exemple: variation **horizontale** `H[i,j] = X[i,j+1] – X[i,j]`



X



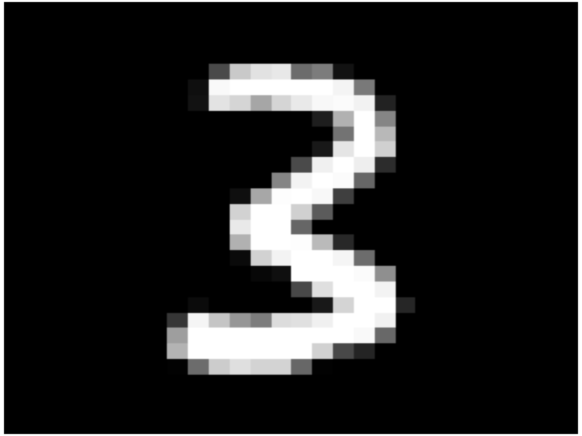
H

`H[14,14] = X[14,15] – X[14,14]`

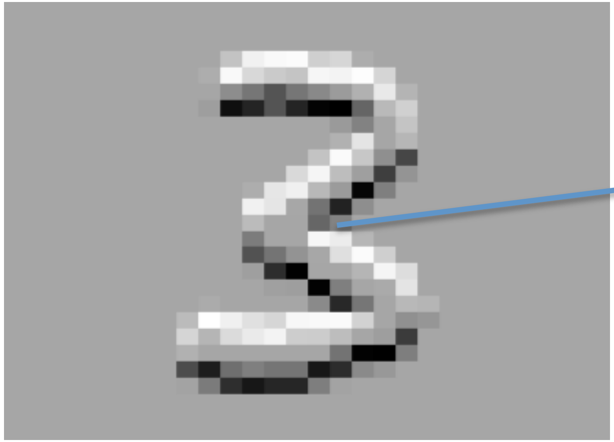
Gradient d’image

- Pour détecter si un pixel est sur la frontière d’un contour, on peut regarder la valeur relative des pixels autour de ce pixel

- Exemple: variation **verticale** $V[i,j] = X[i+1,j] - X[i,j]$



X



V

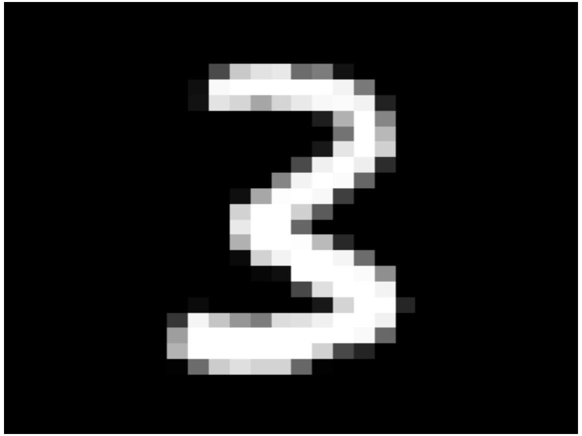
$V[14,14] = X[15,14] - X[14,14]$

Gradient d'image

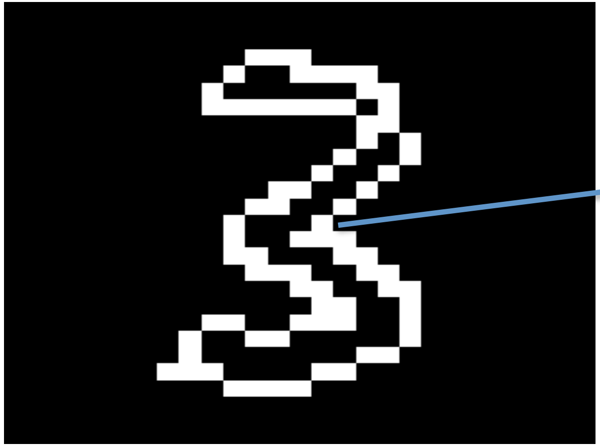
- Un pixel ferait partie d'un contour si la somme des variations (positive ou négative) horizontale et verticale est élevée

```
E[i,j] = sqrt(V[i,j]**2 + H[i,j]**2)
```

- On applique un seuil pour déterminer si contour ou pas



X



E[14,14]>128

E > 128

Hugo Larochelle et Froduald Kabanza

Gradient d'image

- On peut voir le calcul des variations comme des dérivées partielles

- La « fonction » $f(a,b)$ serait la valeur de l'image à la position (a,b)

$$\frac{\partial f(a,b)}{\partial b} = \lim_{\Delta \to 0} \frac{f(a, b+\Delta) - f(a,b)}{\Delta} \approx \underbrace{\texttt{X[i,j+1] - X[i,j]}}_{\Delta = 1} \texttt{ = H[i,j]}$$

$$\frac{\partial f(a,b)}{\partial a} = \lim_{\Delta \to 0} \frac{f(a+\Delta, b) - f(a,b)}{\Delta} \approx \overbrace{\texttt{X[i+1,j] - X[i,j]}}^{\Delta = 1} \texttt{ = V[i,j]}$$

Gradient d'image

- Si `H[i,j]` et `V[i,j]` sont les dérivées partielles de l'image, alors

```
G[i,j,:] = [ H[i,j], V[i,j] ]
```

  est le **gradient de l'image**, à la position `(i,j)`

- On peut visualiser ce gradient (vecteur) à chaque pixel

Champ de vecteurs gradient

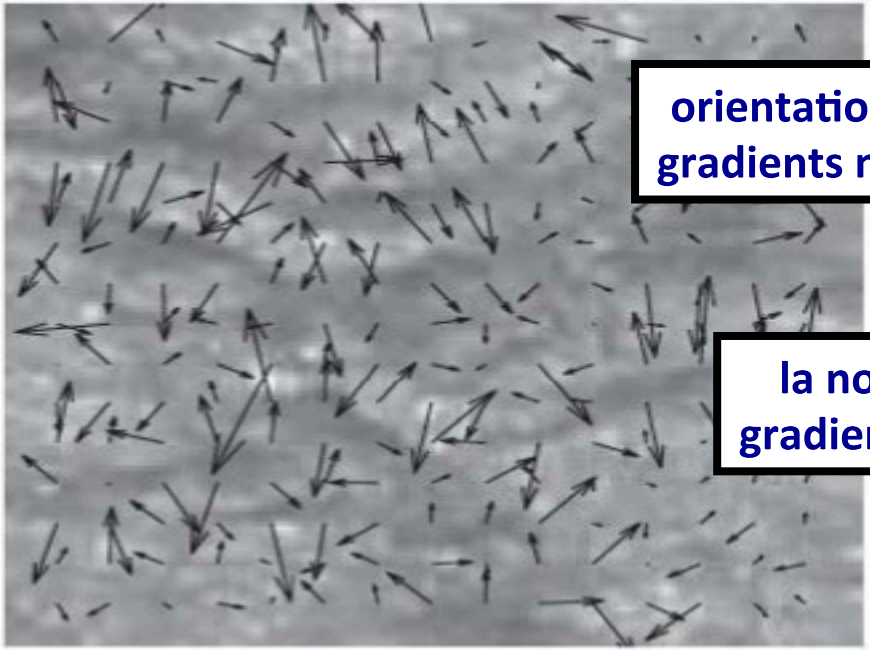


image X

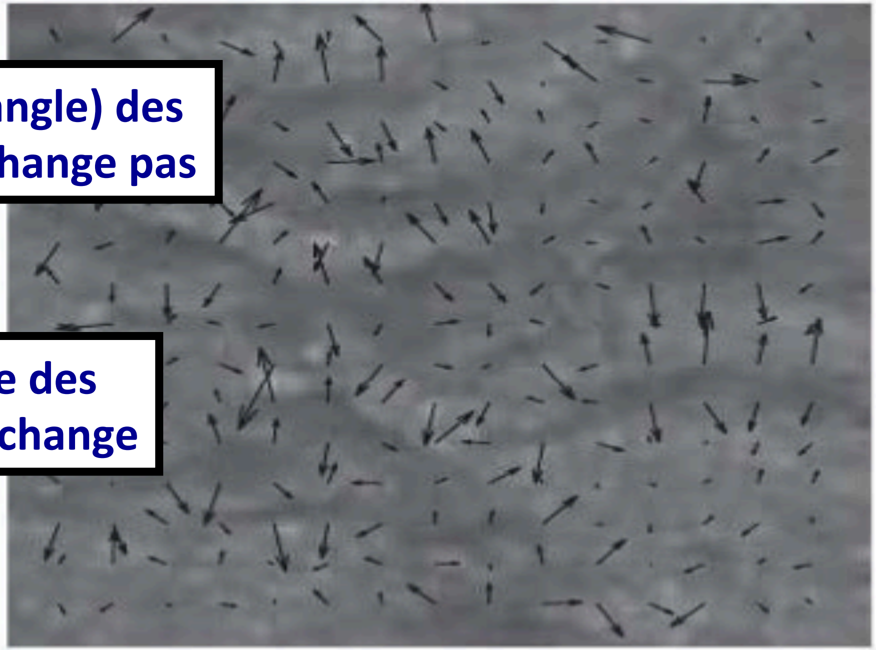


image X avec moins d'illumination

orientation (angle) des gradients ne change pas

la norme des gradients change

Corrélation 2D

- Le calcul des tableaux H et V peut être vu comme l'application d'une **corrélation 2D**

symbole mathématique de la corrélation

0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0

X

$\star$

0	0.5
0.5	0

W

filtre ou **noyau**

=

0	128	128	0
0	128	128	0
0	255	0	0
255	0	0	0

$X \star W$

Corrélation 2D

- Le calcul des tableaux H et V peut être vu comme l'application d'une **corrélation 2D**
- On calcule le résultat C d'une corrélation d'un **filtre** ou **noyau** W de taille h par w sur une image X comme suit

```
def correlation(X,W):
  h,w = W.shape
  C = zeros((X.shape[0]-h+1,X.shape[1]-w+1))
  for i in range(X.shape[0]-h+1):
    for j in range(X.shape[1]-w+1):
      C[i,j] = sum(X[i:i+h,j:j+w] * W)
  return C
```

Corrélation 2D

- Calculer H est l'équivalent de faire une corrélation avec le filtre
`W = array([[-1,1]])`

$$
\underbrace{\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 0 & 0 & 0 \\ 255 & 0 & 0 & 0 & 0 \end{bmatrix}}_{X}
\star
\underbrace{\begin{bmatrix} -1 & 1 \end{bmatrix}}_{W}
=
\underbrace{\begin{bmatrix} 0 & 255 & -255 & 0 \\ 0 & 255 & -255 & 0 \\ 0 & 255 & -255 & 0 \\ 255 & -255 & 0 & 0 \\ -255 & 0 & 0 & 0 \end{bmatrix}}_{X \star W}
$$

Corrélation 2D

- Calculer V est l'équivalent de faire une corrélation avec le filtre
  `W = array([[-1],[1]])`

$$\underbrace{\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 0 & 0 & 0 \\ 255 & 0 & 0 & 0 & 0 \end{bmatrix}}_{X} \star \underbrace{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 255 & -255 & 0 & 0 \\ 255 & -255 & 0 & 0 & 0 \end{bmatrix}}_{X \star W}$$

Corrélation 2D

- Afin d'appliquer le filtre à toutes les positions dans l'image, on ajoute parfois les zéros nécessaires autour de l'image (*zero padding*)

0	0	0	0	0
0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0
0	0	0	0	0

X

$\star$

-1
1

W

=

0	0	255	0	0
0	0	0	0	0
0	0	0	0	0
0	255	-255	0	0
255	-255	0	0	0
-255				

$X \star W$

Convolution 2D

- Une opération liée à la corrélation et fréquemment utilisée en vision par ordinateur est la **convolution 2D**
- Comme la corrélation, mais le point de référence des indexes du filtre débute à la dernière rangée et colonne

symbole mathématique de la convolution

$$\underbrace{\begin{bmatrix} 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 \\ 0 & 255 & 0 & 0 & 0 \\ 255 & 0 & 0 & 0 & 0 \end{bmatrix}}_{X} * \underbrace{\begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} 0 & 128 & 128 & 0 \\ 0 & 128 & 128 & 0 \\ 0 & 255 & 0 & 0 \\ 255 & 0 & 0 & 0 \end{bmatrix}}_{X * W}$$

Convolution 2D

- Une opération liée à la corrélation et fréquemment utilisée en vision par ordinateur est la **convolution 2D**
- Est équivalent à faire une corrélation avec un nouveau filtre dont on a inversé l'ordre des rangées et des colonnes

```
def convolution(X,W):
    return correlation(X,W[::-1,::-1])
```

- Le résultat peut parfois être le même
  - par exemple si le filtre est symétrique horizontalement et verticalement