

# **IFT 615 – Intelligence artificielle**

## **Vision par ordinateur**

Hugo Larochelle

Département d'informatique

Université de Sherbrooke

<http://www.dmi.usherb.ca/~larocheh/cours/ift615.html>

# Sujets couverts

- Opérations bas niveau sur les images
  - ◆ détection de contour
  - ◆ calcul de gradients d'image
- Reconnaissance d'objets
  - ◆ à base de caractéristiques (histogramme de gradients)
  - ◆ à l'aide d'un réseau de neurones à convolution

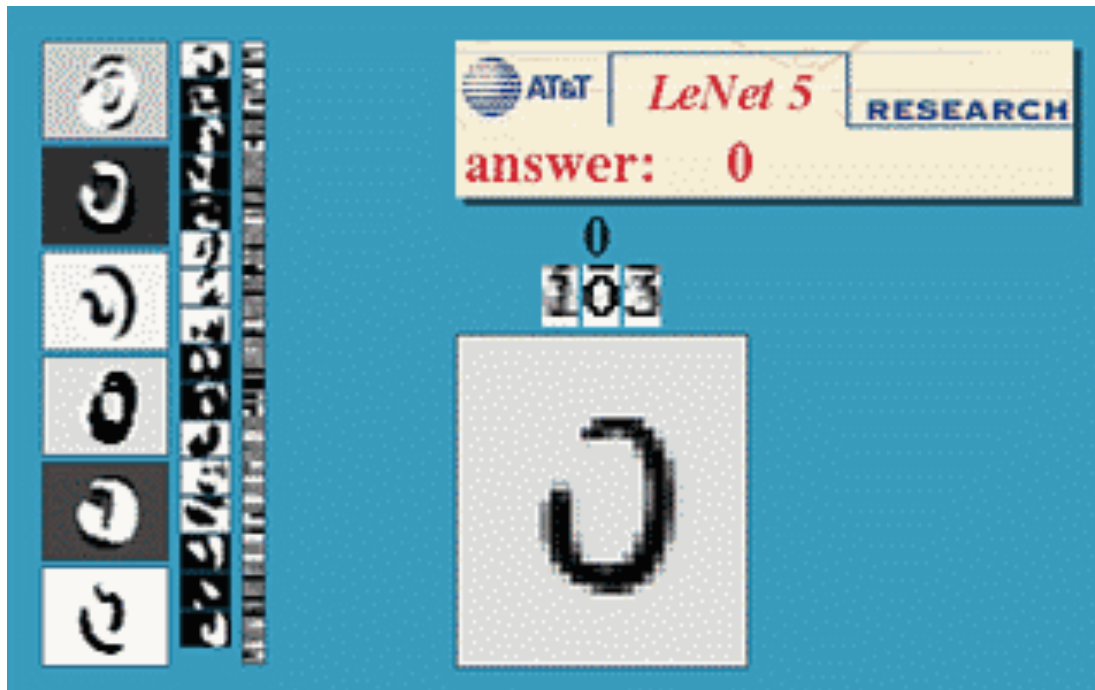
# Mise en situation

- La vue est un sens très utile à la survie d'un organisme
  - ◆ apporte beaucoup d'information sur son environnement (nourriture, prédateur, etc.)
- Presque toutes les créatures intelligentes sont dotées de vision
- Chez l'humain  $\approx 30\%$  du cerveau sert à la vision
  - ◆ pour l'ouïe, c'est  $\approx 8\%$
  - ◆ pour le touché, c'est  $\approx 3\%$
- Ça donne une idée de la complexité de la tâche à résoudre...

# Mise en situation

- Applications liées à la vision par ordinateur

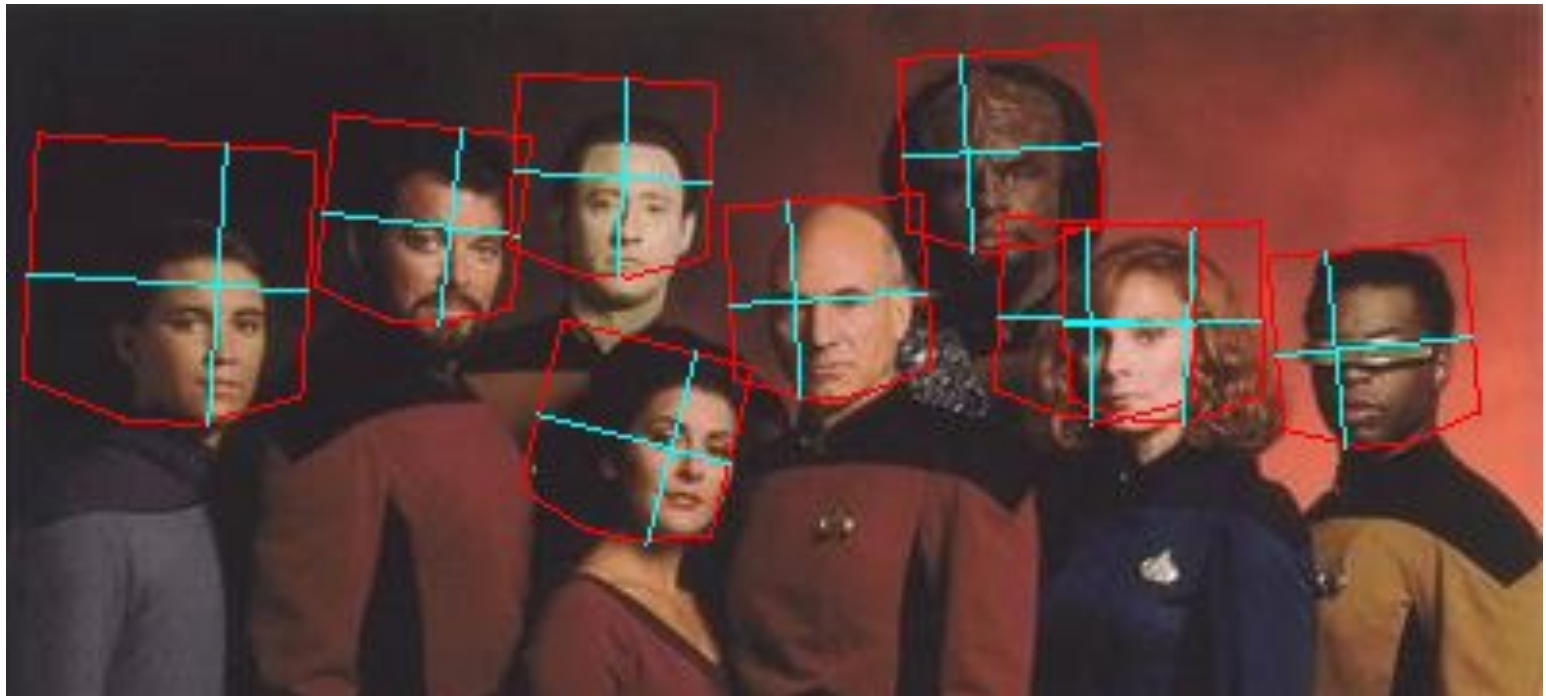
## Reconnaissance de caractères



# Mise en situation

- Applications liées à la vision par ordinateur

## Détection de visages



# Mise en situation

- Applications liées à la vision par ordinateur

## Recherche d'images par contenu

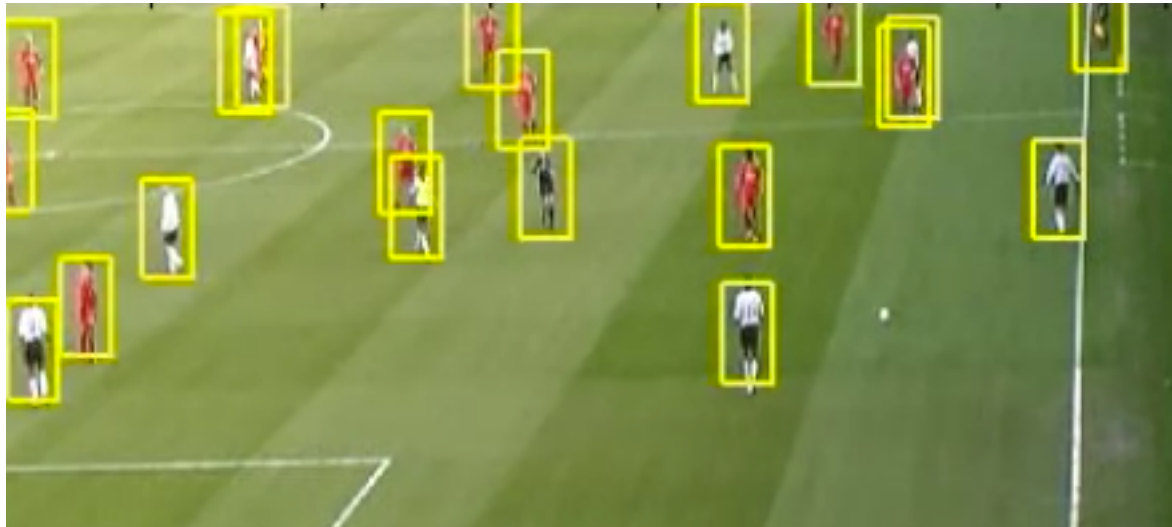


<http://images.google.com/>

# Mise en situation

- Applications liées à la vision par ordinateur

## Suivi d'objets



<http://www.youtube.com/watch?v=fRowYlxKt7s>

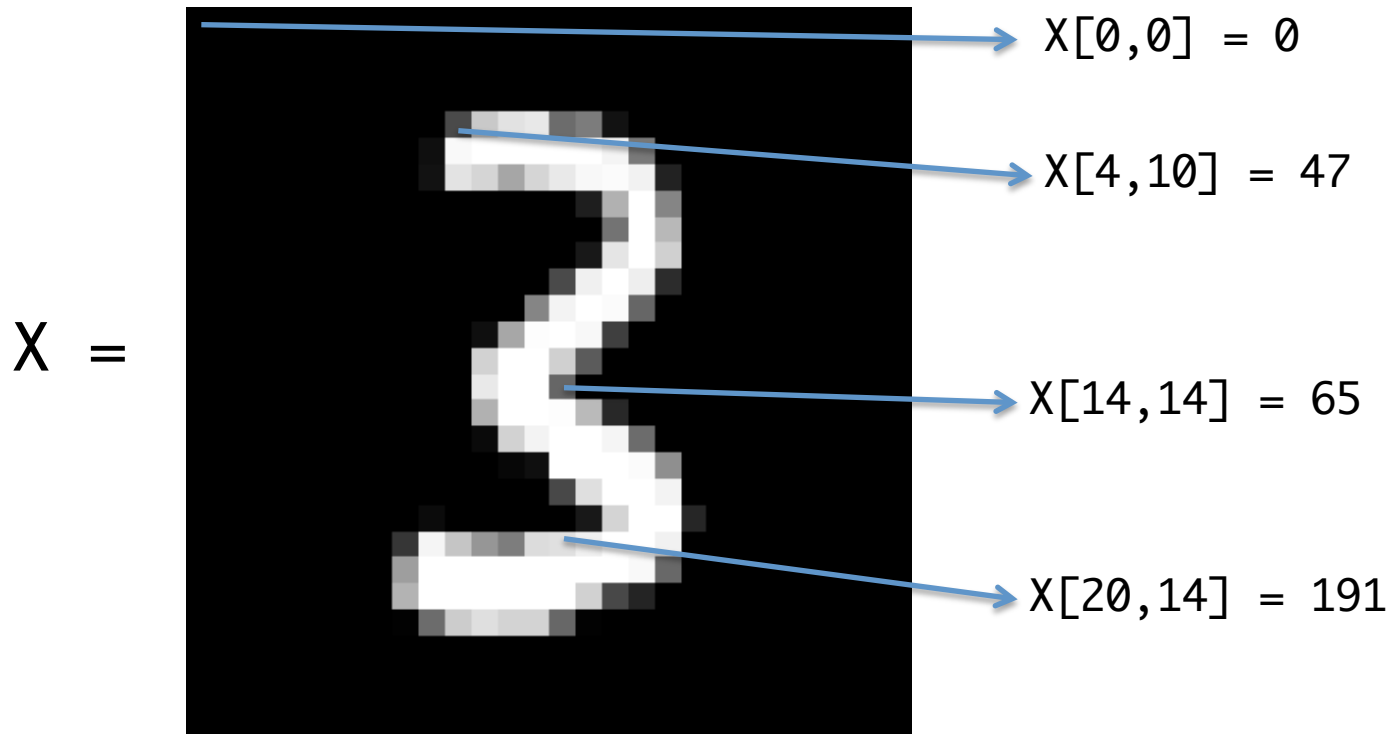
# Dans ce cours...

- On va voir comment on manipule des images
  - ◆ quelle représentation de base utiliser
  - ◆ quel genre de prétraitements sont utiles
- L'objectif est d'avoir une vue d'ensemble des approches suivies en vision par ordinateur
- On va discuter des concepts fréquemment utilisés en vision
  - ◆ convolution
  - ◆ gradients d'image
  - ◆ histogramme
  - ◆ « pooling »



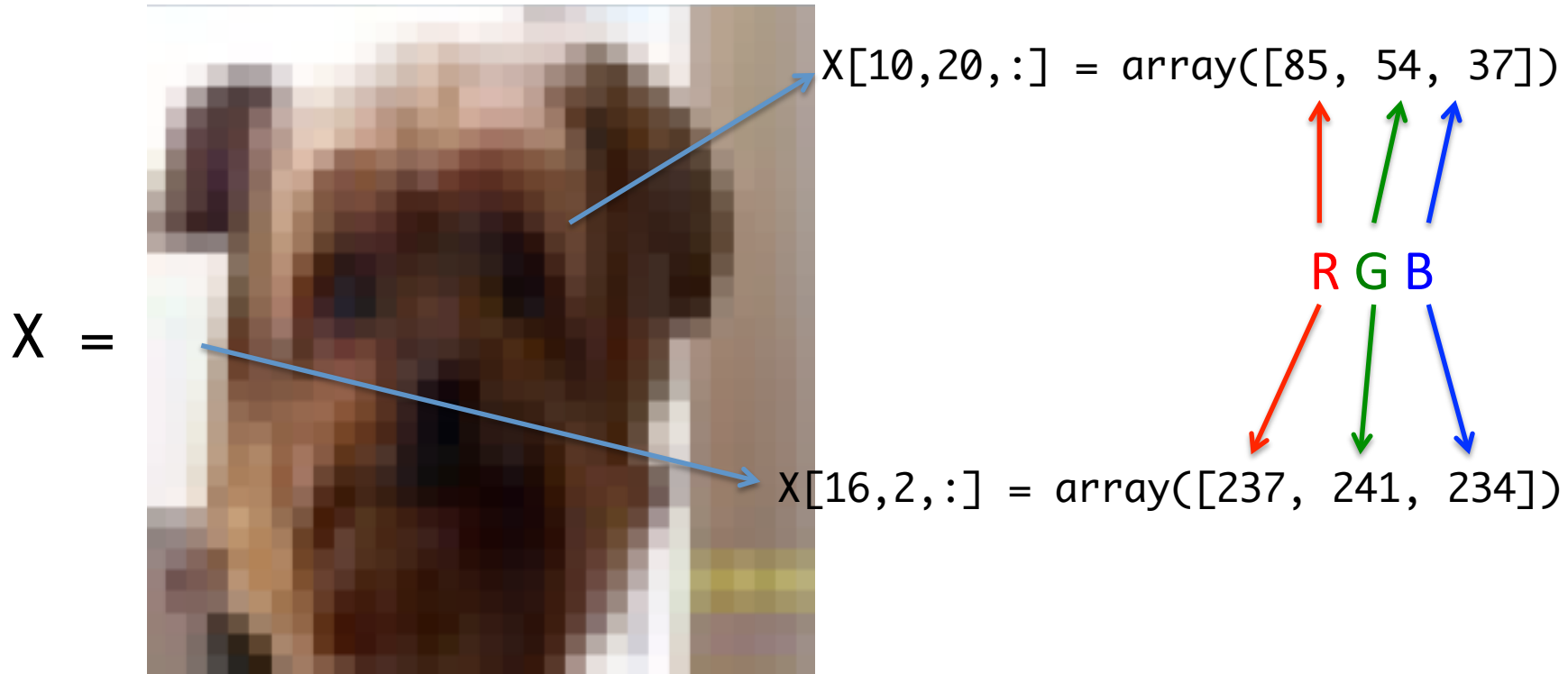
# Représentation brute d'une image

- Image en niveau de gris: tableau à 2D de pixels, entiers positifs de 8 bits



# Représentation brute d'une image

- Image en couleur: tableau à 3D de pixels RGB, entiers positifs de 8 bits



# Opérations bas niveau sur les images

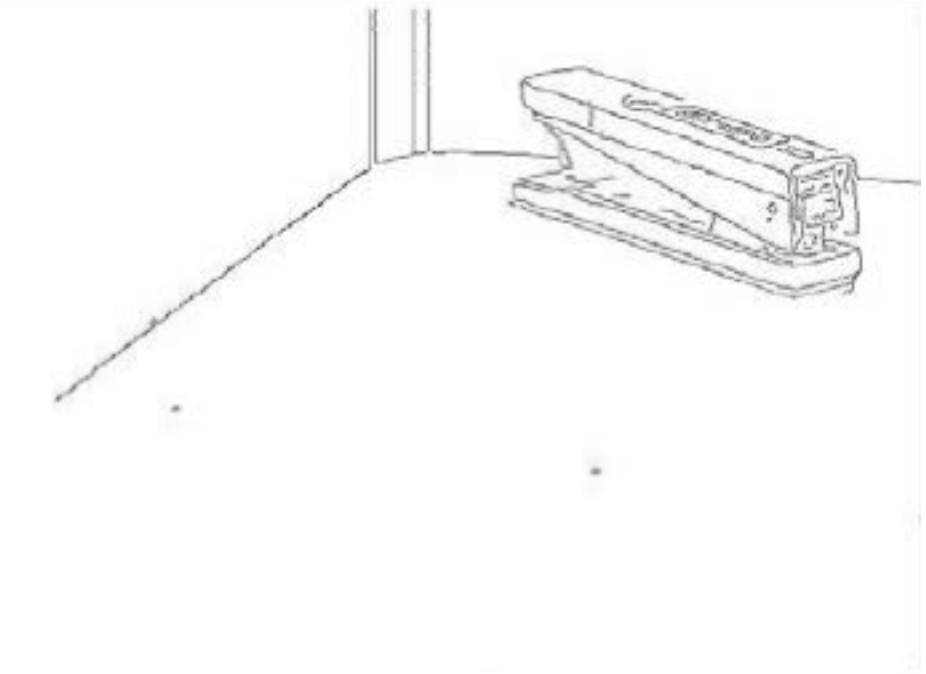
- La représentation sous forme de pixels a des désavantages
  - ◆ elle est lourde, c.-à-d. coûteuse en mémoire
    - » 1024x1024 pixels de 8 bits (en niveau de gris) = 1 MB / image
    - » 1024x1024 pixels de 24bits (canaux RGB) = 3 MB / image
  - ◆ elle contient plus d'information qu'on en a besoin
    - » 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

# Détection de contour

- Un contour est une changement soudain dans l'intensité/couleur de pixels adjacents



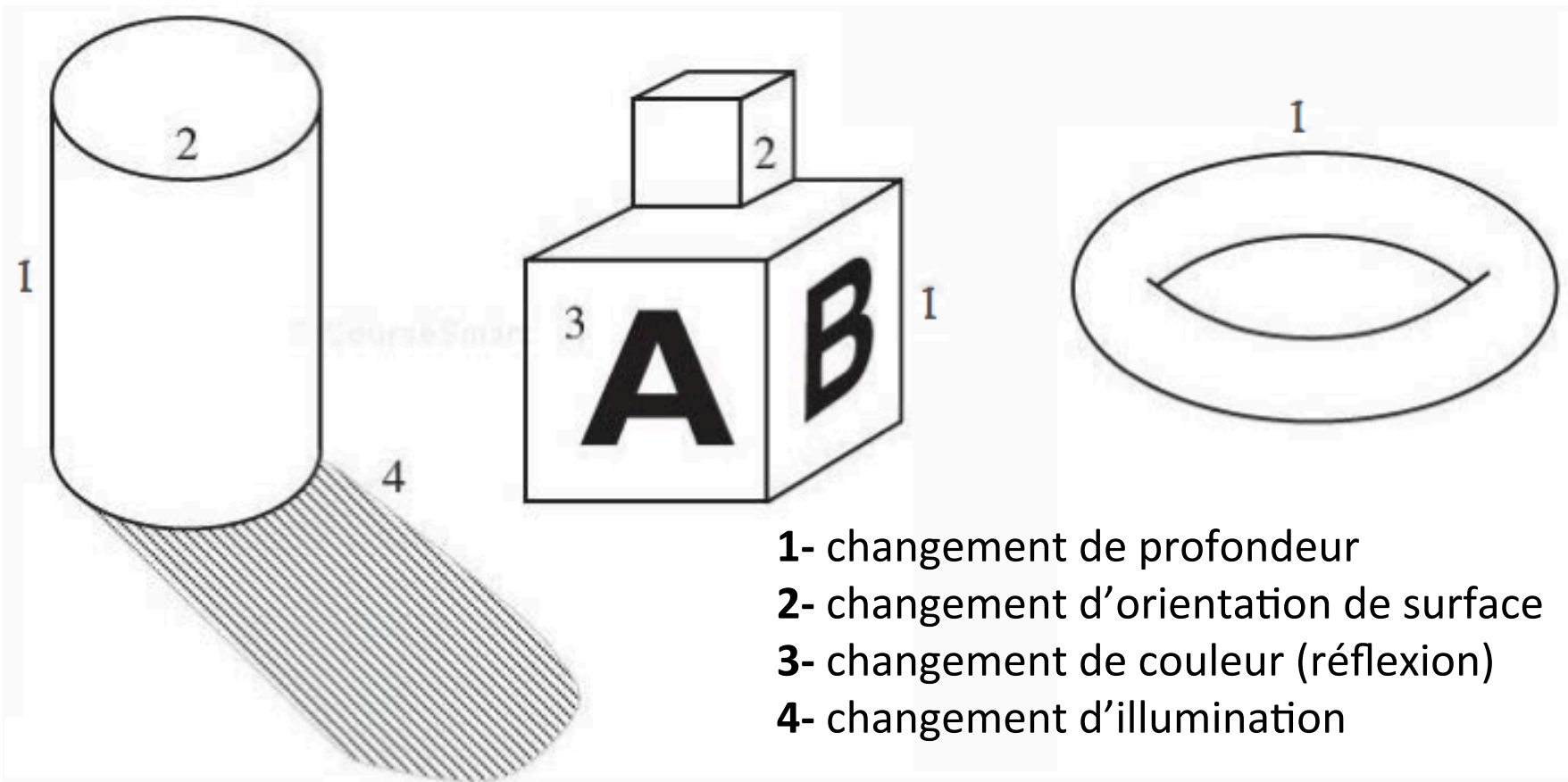
image originale



extraction des contours

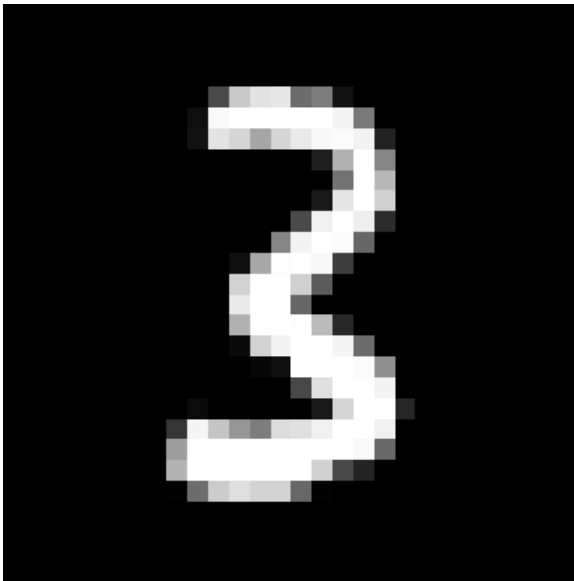
# Détection de contour

- Qu'est-ce qui cause des contours?

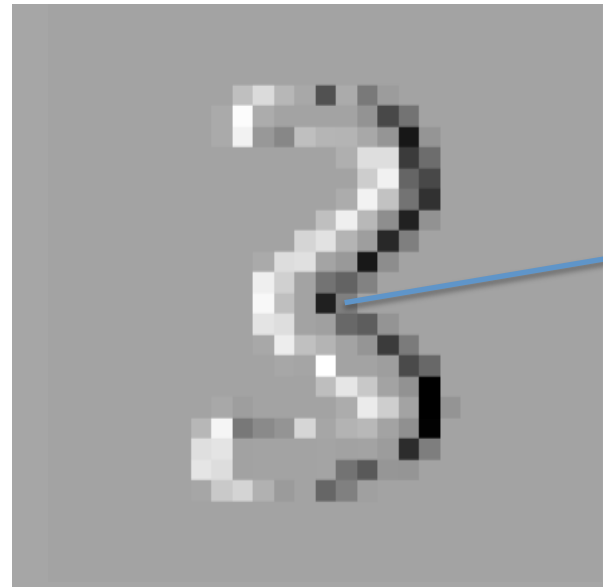


# Gradient d'image

- 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

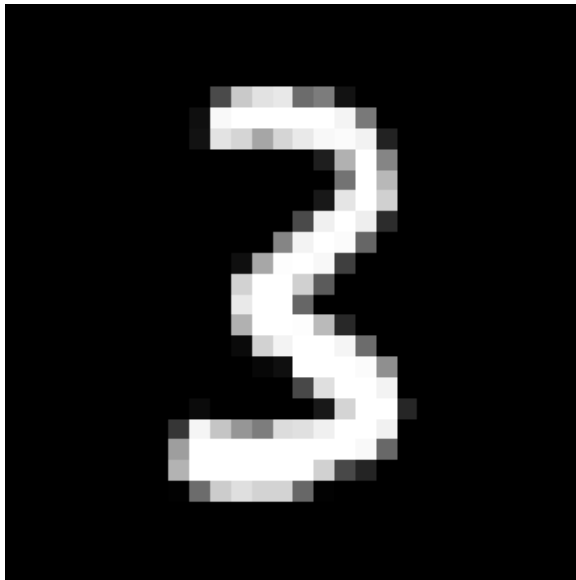


H

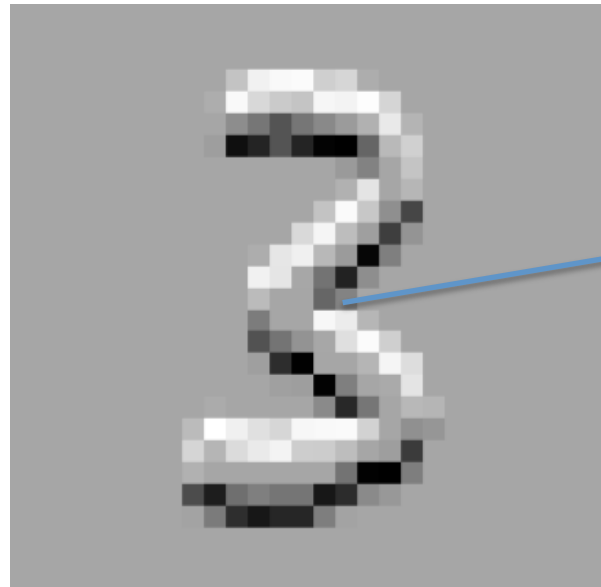
$$H[14, 14] = X[14, 15] - X[14, 14]$$

# 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 **vertical**  $V[i, j] = X[i+1, j] - X[i, j]$



X



V

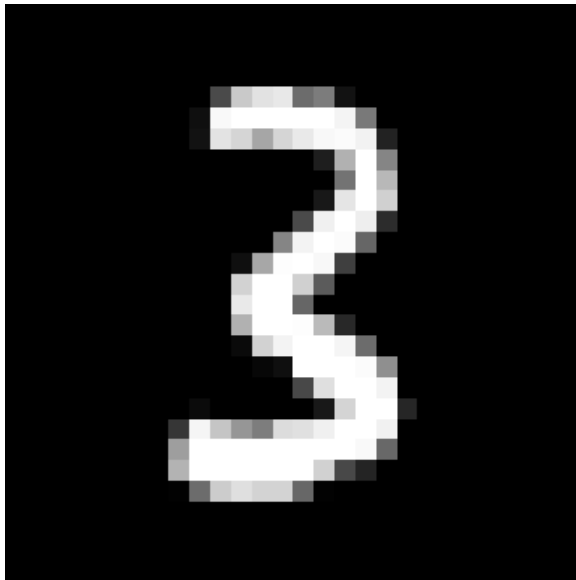
$V[14, 14] =$   
 $X[15, 14] -$   
 $X[14, 14]$

# Gradient d'image

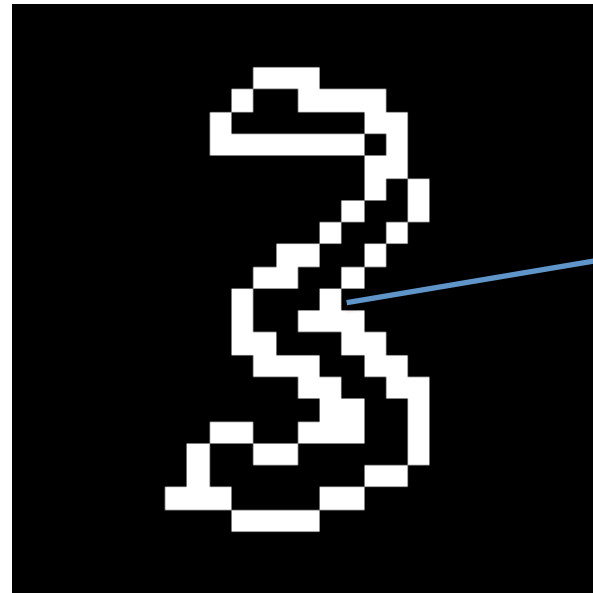
- Un pixel ferait partie d'un contour si la somme des variations (positive ou négative) horizontale et verticale est élevée

$$E[i, j] = \text{sqrt}(V[i, j]**2 + H[i, j]**2)$$

- On applique un seuil pour déterminer si contour ou pas



X



$E > 0.5$

$E[14, 14] > 0.5$



# 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  $(a, b)$

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

$$\Delta = 1$$

$$\frac{\partial f(a, b)}{\partial a} = \lim_{\Delta \rightarrow 0} \frac{f(a + \Delta, b) - f(a, b)}{\Delta} \approx \underbrace{x[i+1, j] - x[i, j]}_{\Delta = 1} = V[i, j]$$

# Gradient d'image

- Si  $I$  et  $J$  sont les dérivées partielles de l'image, alors

$$G[i, j, :] = [ I[i, j], J[i, j] ]$$

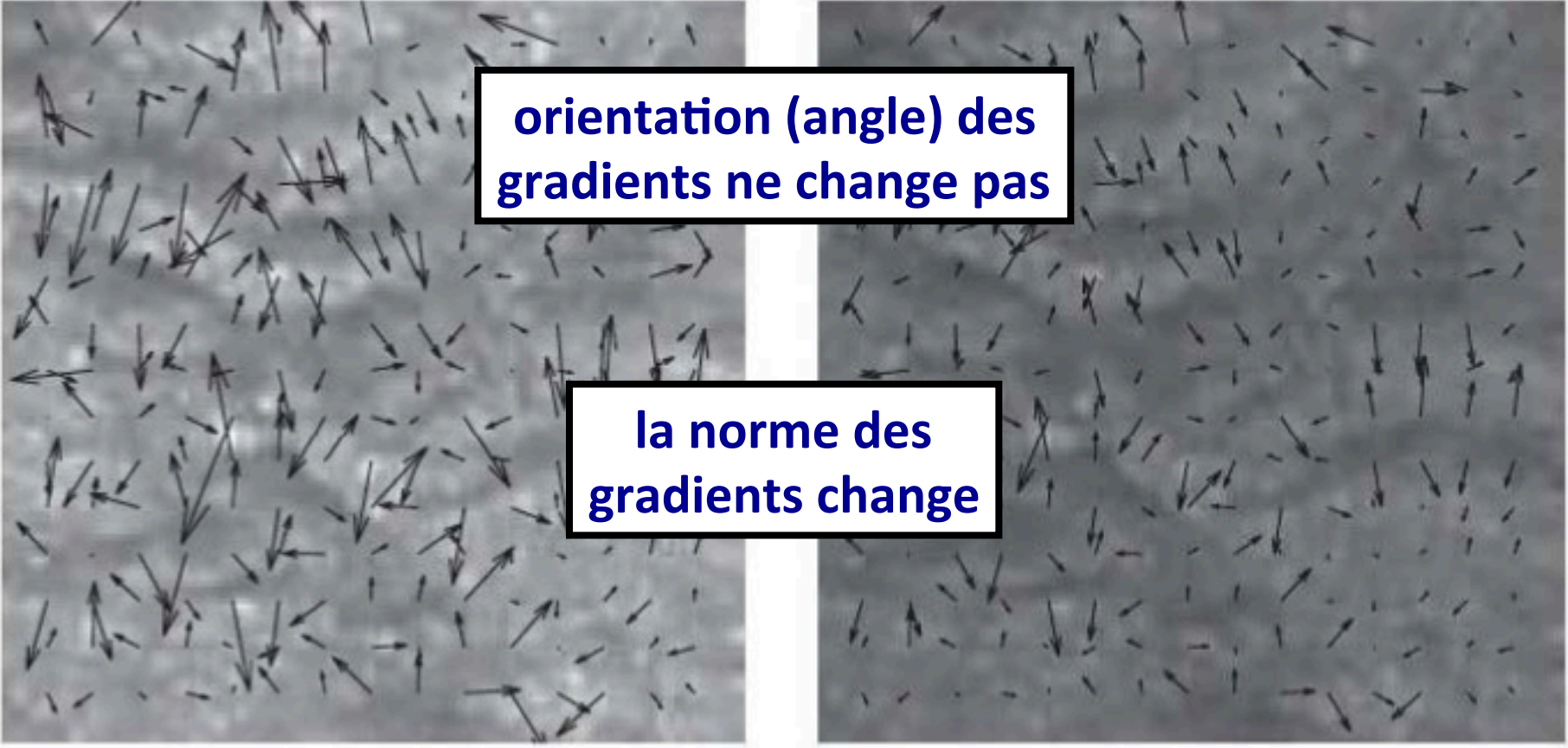
est le **gradient de l'image**, à la position  $(i, j)$

- La détection des contours vue précédemment calculait donc la norme euclidienne de ces gradients

$$E[i, j] = \sqrt{I[i, j]^2 + J[i, j]^2} = \underbrace{\sqrt{\text{sum}(G[i, j, :]^2)}}_{\text{norme du vecteur } G[i, j, :]}$$

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

# Champ de vecteurs gradient



**orientation (angle) des  
gradients ne change pas**

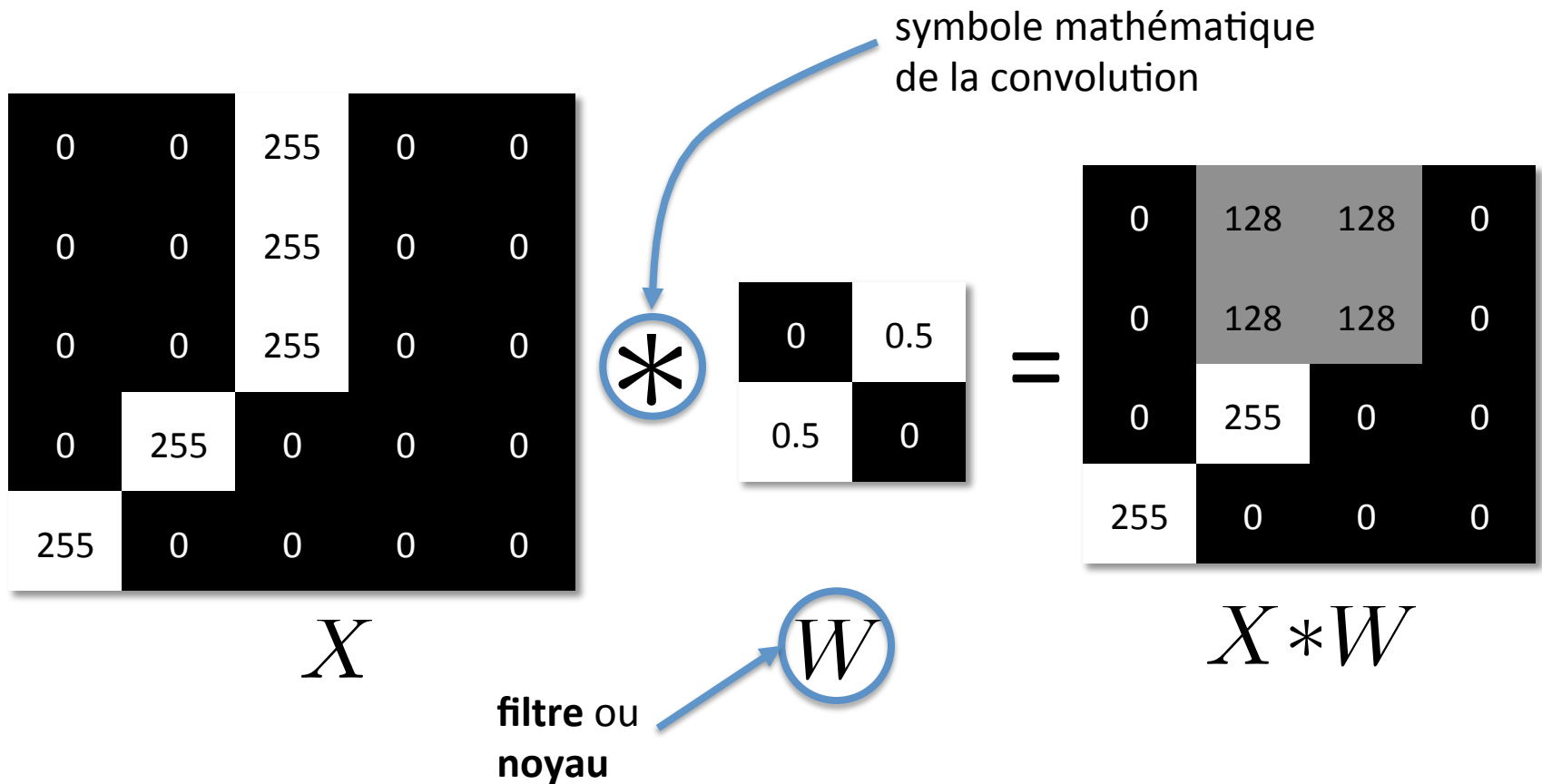
**la norme des  
gradients change**

image X

image X avec moins d'illumination

# Convolution

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



# Convolution

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

```
def convolution(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
```

# Convolution

- Calculer  $H$  est l'équivalent de faire une convolution avec le filtre  $W = \text{array}([[[-1, 1]]])$

The diagram illustrates a 1D convolution operation. On the left is a 5x5 input matrix  $X$  with values:

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

Below the matrix is the label  $X$ . In the middle is a 1x2 kernel matrix  $W$  with values  $[-1, 1]$ . Below the kernel is the label  $W$ . To the right of the kernel is an equals sign. On the far right is the resulting 5x5 output matrix  $X * W$  with values:

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

Below the output matrix is the label  $X * W$ . The operation is represented by a convolution symbol  $*$  between  $X$  and  $W$ .

# Convolution

- Calculer  $V$  est l'équivalent de faire une convolution avec le filtre  $W = \text{array}([[ -1], [ 1]])$

The diagram illustrates a 1D convolution operation. It shows three matrices:  $X$ ,  $W$ , and the result  $X * W$ .

Matrix  $X$  (Input) is a 5x5 grid:

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

Matrix  $W$  (Filter) is a 2x1 grid:

-1
1

The result matrix  $X * W$  is a 5x5 grid:

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

# Convolution

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

The diagram illustrates a 1D convolution operation. It shows an input vector  $X$  of size 7, which is padded to size 9 (adding zeros at both ends). This padded vector is then convolved with a kernel  $W$  of size 2 to produce an output vector  $X * W$  of size 8.

**Input Vector  $X$  (7x1):**

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

**Kernel  $W$  (2x1):**

-1
1

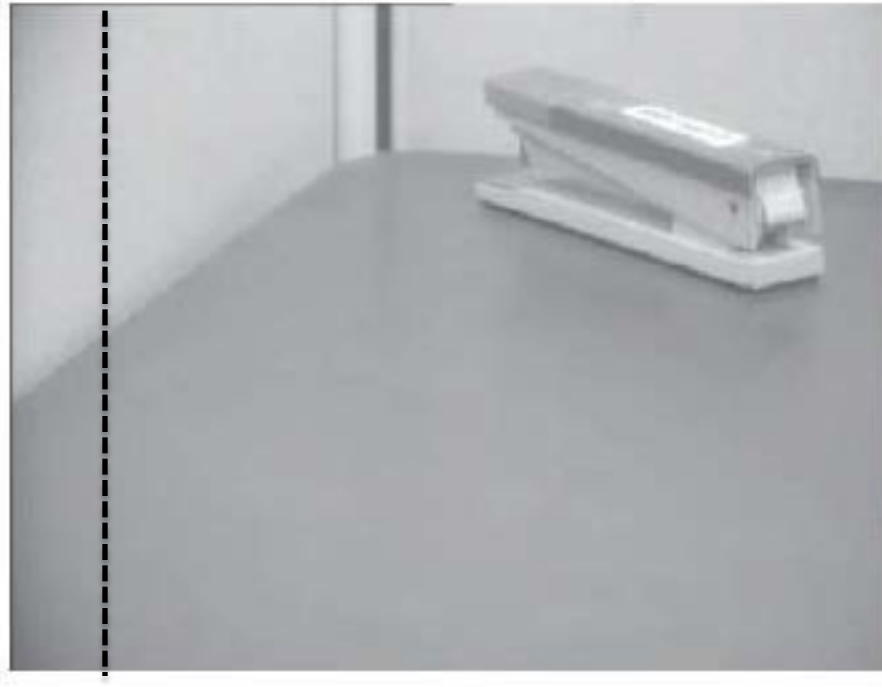
**Output Vector  $X * W$  (8x1):**

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				



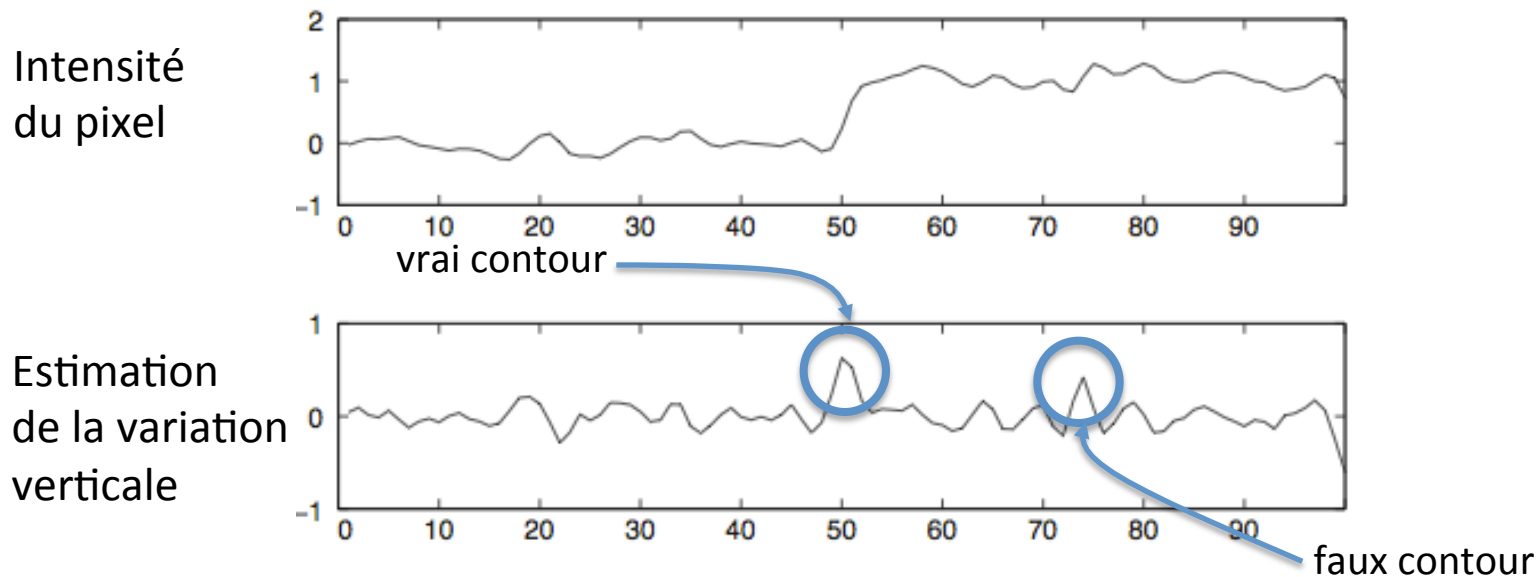
# Bruit dans le calcul du gradient d'image

- Sur de vrais images, l'estimation des variations sera bruitée



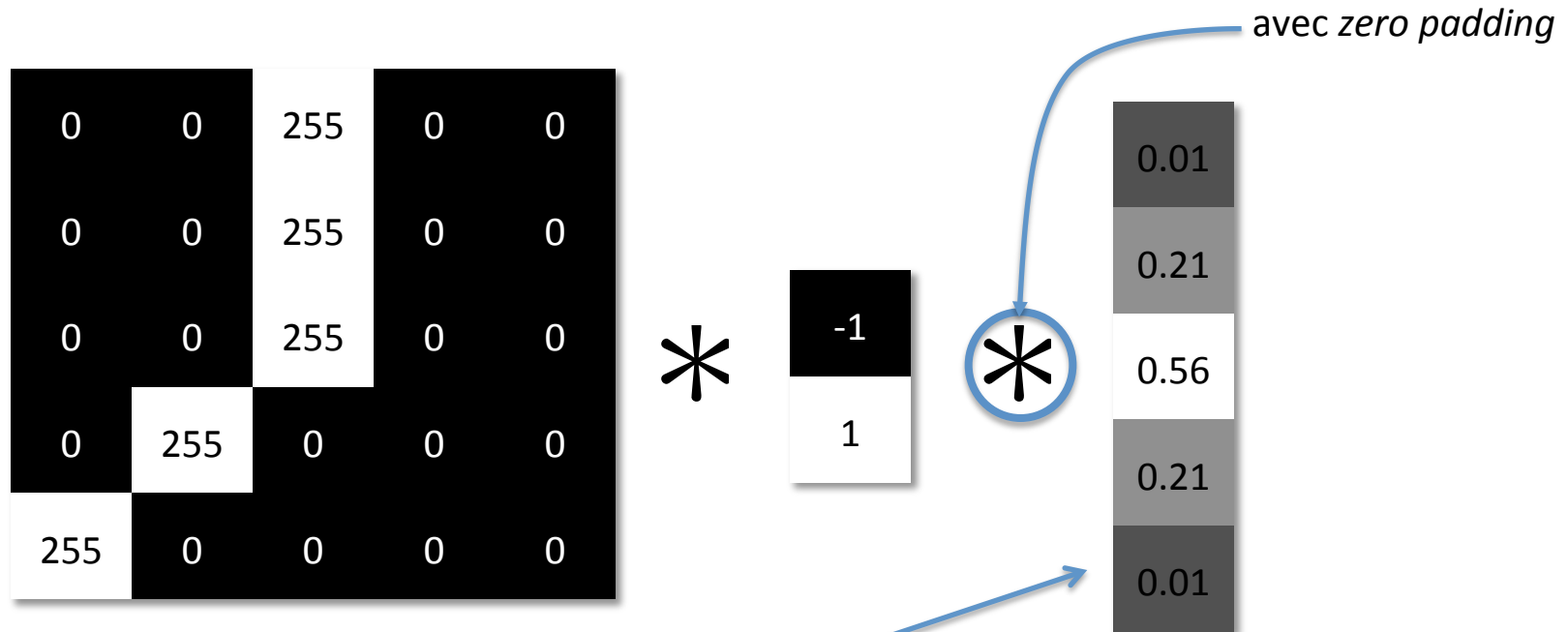
# Bruit dans le calcul du gradient d'image

- Sur de vrais images, l'estimation des variations sera bruitée



# Bruit dans le calcul du gradient d'image

- Pour éliminer la détection de ces faux contours, on applique une deuxième convolution pour **lisser** le résultat



- Le filtre utilisé est appelé **filtre gaussien**

# Bruit dans le calcul du gradient d'image

- Pour éliminer la détection de ces faux contours, on applique une deuxième convolution pour **lisser** le résultat

0.01	$= \exp(-2^2) / Z$
0.21	$= \exp(-1^2) / Z$
0.56	$= \exp(-0^2) / Z$
0.21	$= \exp(-1^2) / Z$
0.01	$= \exp(-1^2) / Z$

constante de normalisation

$$Z = \exp(-2^2) + \exp(-1^2) + \exp(-0^2) + \exp(-1^2) + \exp(-2^2)$$

# Bruit dans le calcul du gradient d'image

- Pour éliminer la détection de ces faux contours, on applique une deuxième convolution pour **lisser** le résultat

0.01	= $\exp(-2^2) / Z$
0.21	= $\exp(-1^2) / Z$
0.56	= $\exp(-0^2) / Z$
0.21	= $\exp(-1^2) / Z$
0.01	= $\exp(-1^2) / Z$

## Formule générale du filtre gaussien

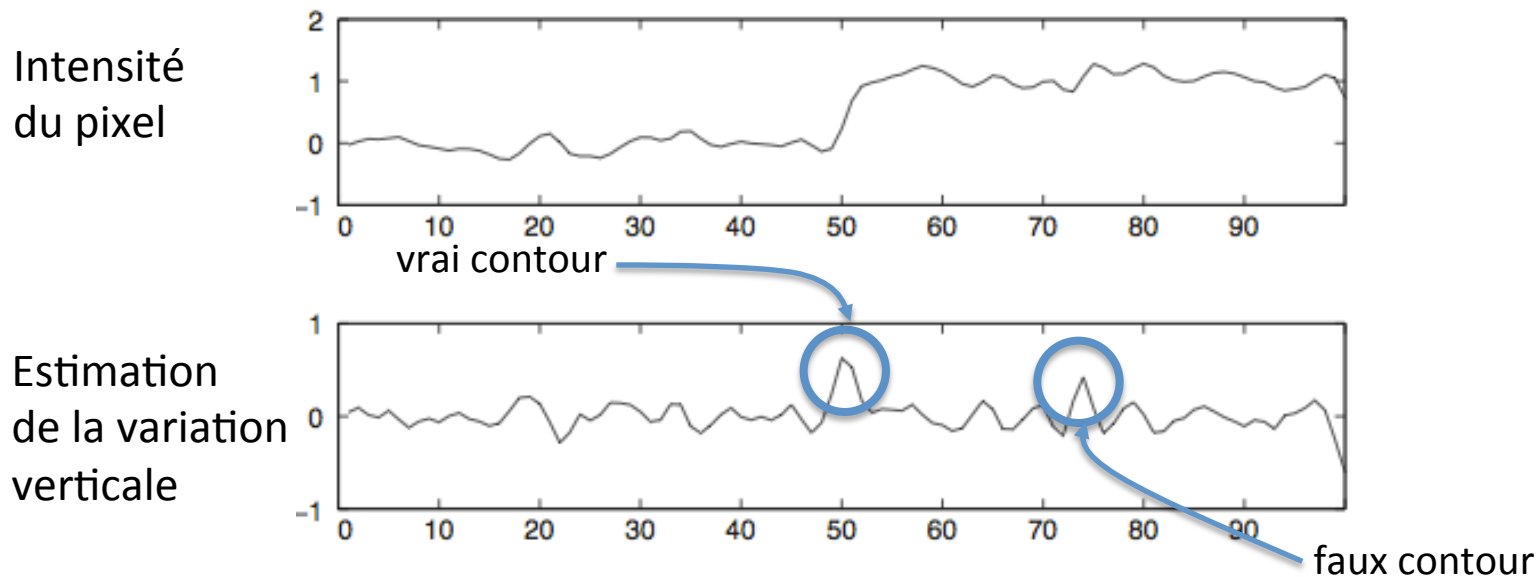
$$W[i, j] = \exp(-d(i, j)^2 / \sigma^2) / Z$$

$d(i, j)$  = distance p/r au centre du filtre

$\sigma$  = paramètre de lissage  
(plus il est grand, plus on lisse)

# Bruit dans le calcul du gradient d'image

- Sur de vrais images, l'estimation des variations sera bruitée



# Si on va plus loin...

- L'estimation des gradients tel que présentée ( $X[i, j+1] - X[i, j]$ ) peut être améliorée
  - ◆ voir les filtres de Sobel (*Sobel operator*)  
[http://en.wikipedia.org/wiki/Sobel\\_operator](http://en.wikipedia.org/wiki/Sobel_operator)
- La détection des contours à l'aide d'un simple seuil peut être améliorée
  - ◆ voir le filtre de Canny (*Canny edge detector*)  
[http://en.wikipedia.org/wiki/Canny\\_edge\\_detector](http://en.wikipedia.org/wiki/Canny_edge_detector)
- On peut extraire à partir des contours l'information sur la présence de ligne droites ou de cercles (ex.: un robot qui veut détecter les limites d'une pièce)
  - ◆ [http://en.wikipedia.org/wiki/Hough\\_transform](http://en.wikipedia.org/wiki/Hough_transform)

# Caractéristiques d'images

- En plus de servir à détecter des contours, les gradients d'image peuvent servir à **extraire des caractéristiques** d'une image
- On a vu que l'orientation des gradients ne varient pas en fonction de l'intensité
  - ◆ on pourrait utiliser cette propriété pour obtenir des caractéristiques invariantes p/r à l'intensité d'une image (ex.: l'illumination)

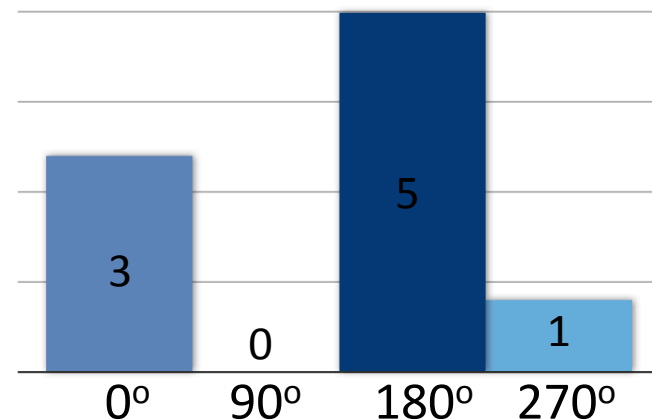
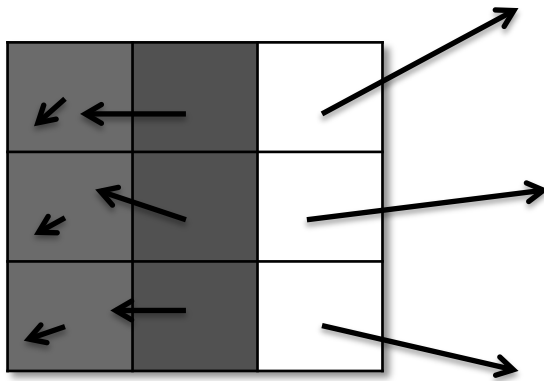


# Histogramme de gradients

- Un type de caractéristiques populaire est l'**histogramme de gradients** (*histogram of gradients* ou **HoG**)
  - ◆ calculer le **champ de gradients** de l'image
  - ◆ partitionner (diviser) l'image en plusieurs **segments** (*cell*)
  - ◆ dans chaque segment, faire un **histogramme** des orientations des gradients contenus dans ce segment
  - ◆ le vecteur de caractéristiques pour l'image est la concaténation de tous ces histogrammes

# Histogramme de gradients

- Pour calculer un histogramme d'orientations
  - ◆ on partitionne les orientations possibles en quelques cases (ex. 4 cases à  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  et  $270^\circ$ , ou 8 cases à  $0^\circ$ ,  $45^\circ$ , ...,  $315^\circ$ )
  - ◆ la valeur de chaque case est le compte du nombre de gradients qui tombent dans chaque case
    - » chaque gradient « vote » pour l'orientation la plus proche

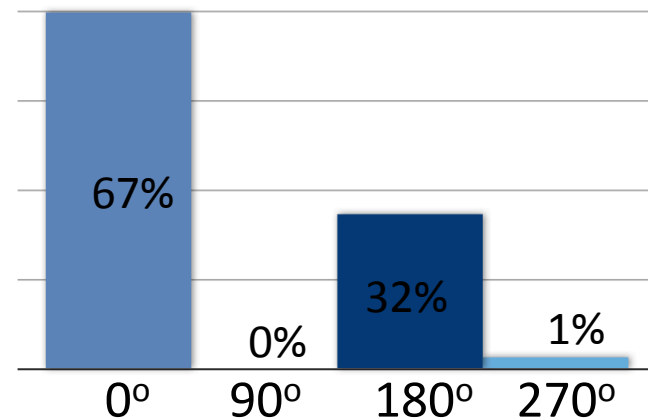
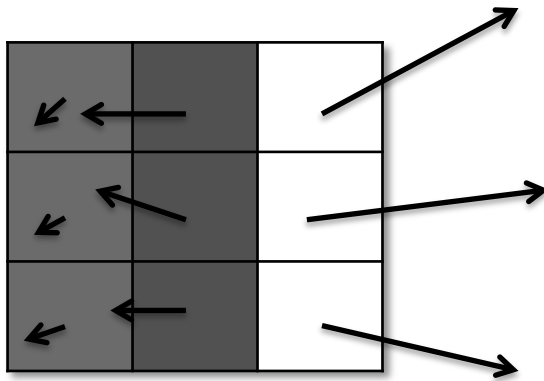


# Histogramme de gradients

- Façon de tenir compte de la grandeur (norme) du gradient
  - ◆ on peut ajouter une case additionnelle pour les gradients dont la norme est sous un seuil donné
  - ◆ on utilise un poids  $C$  du vote d'un gradient  $G[i, j, :]$  comme suit

$$c[i, j] = \text{sqrt}(\text{sum}(G[i, j, :]**2)) / N$$

où  $N$  est la somme des normes dans le segment où se trouve  $G[i, j, :]$



# Histogramme de gradients

- Avoir une case pour les gradient trop petits permet de mieux représenter les régions uniformes (où tous les gradients sont petits)
- Utiliser un vote normalisé par la somme des normes de gradient donne une représentation plus invariante p/r à l'illumination
- Les deux idées peuvent être combinées

# Histogramme de gradients

- Exemple sur une image de personne



Image



Orientation  
histograms

illustre la  
perpendiculaire  
du gradient, pour  
faciliter  
la visualisation

# Histogramme de gradients

- On peut alors traiter le problème de reconnaissance d'objets comme un problème de classification standard en apprentissage automatique
  - ◆ entrée  $x_t$ : représentation HoG d'une image
  - ◆ cible  $y_t$ : présence ( $y_t=1$ ) ou absence ( $y_t=0$ ) d'un objet à reconnaître
- On peut ainsi collecter un ensemble d'entraînement à donner à un algorithme d'apprentissage pour la classification  
(Perceptron, régression linéaire, réseau de neurones, etc.)

# Histogramme de gradients

- Visualisation des orientations importantes, apprises par un classifieur linéaire



Image



Orientation  
histograms

Visualisation des poids du classifieur



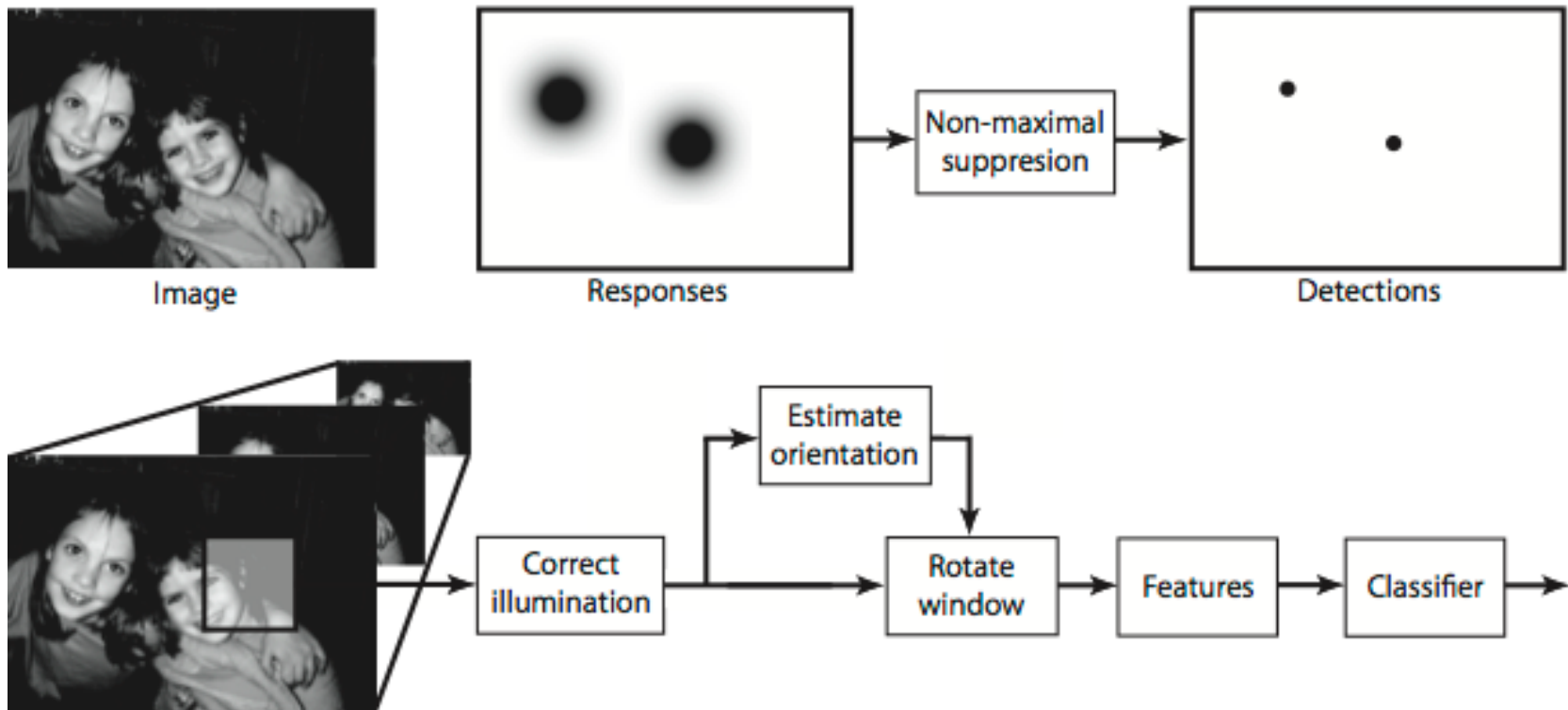
poids  
positifs



poids  
négatifs

# De la classification à la détection

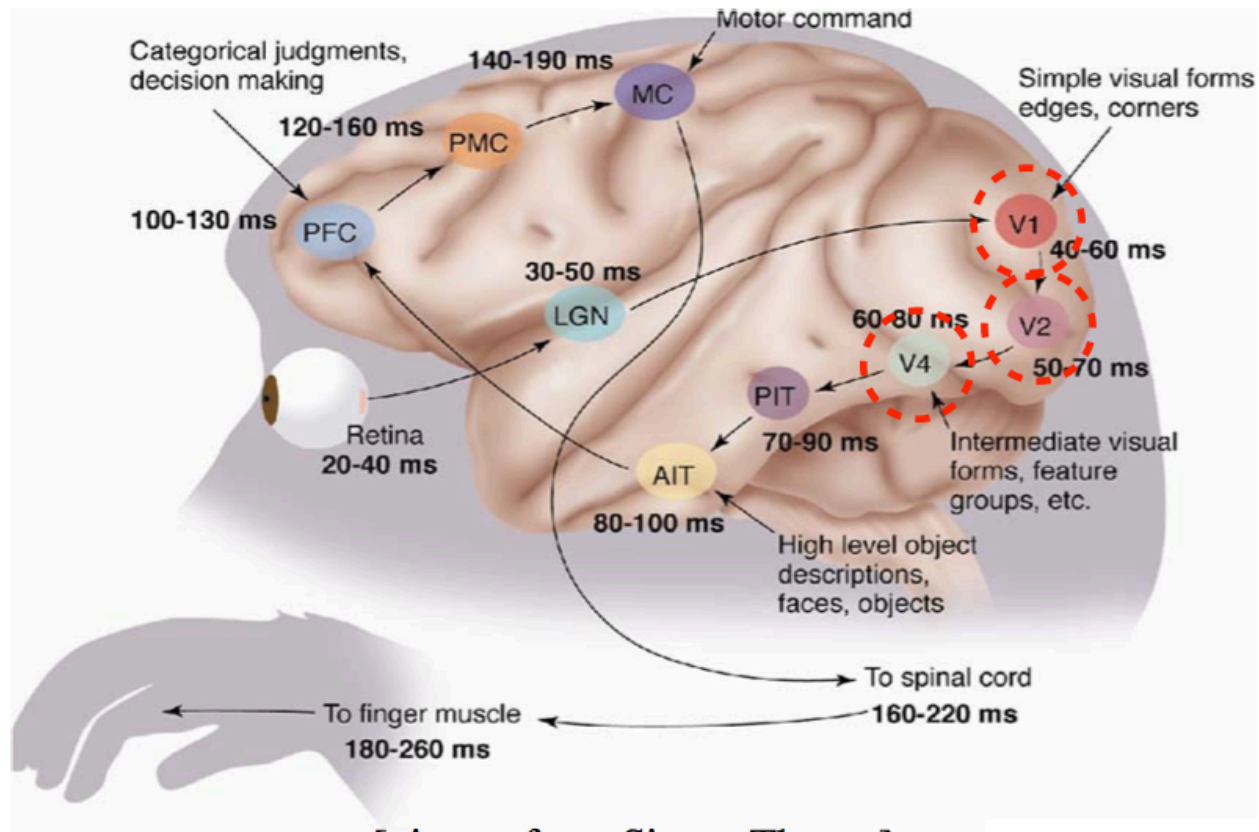
- Quoi faire si ce que l'on cherche n'est pas au centre de l'image?
- **Idée générale:** on applique le même classifieur à plusieurs positions et échelles dans l'image





# Le système visuel humain

- Pourquoi ne pas s'inspirer du cerveau pour faire de la vision!

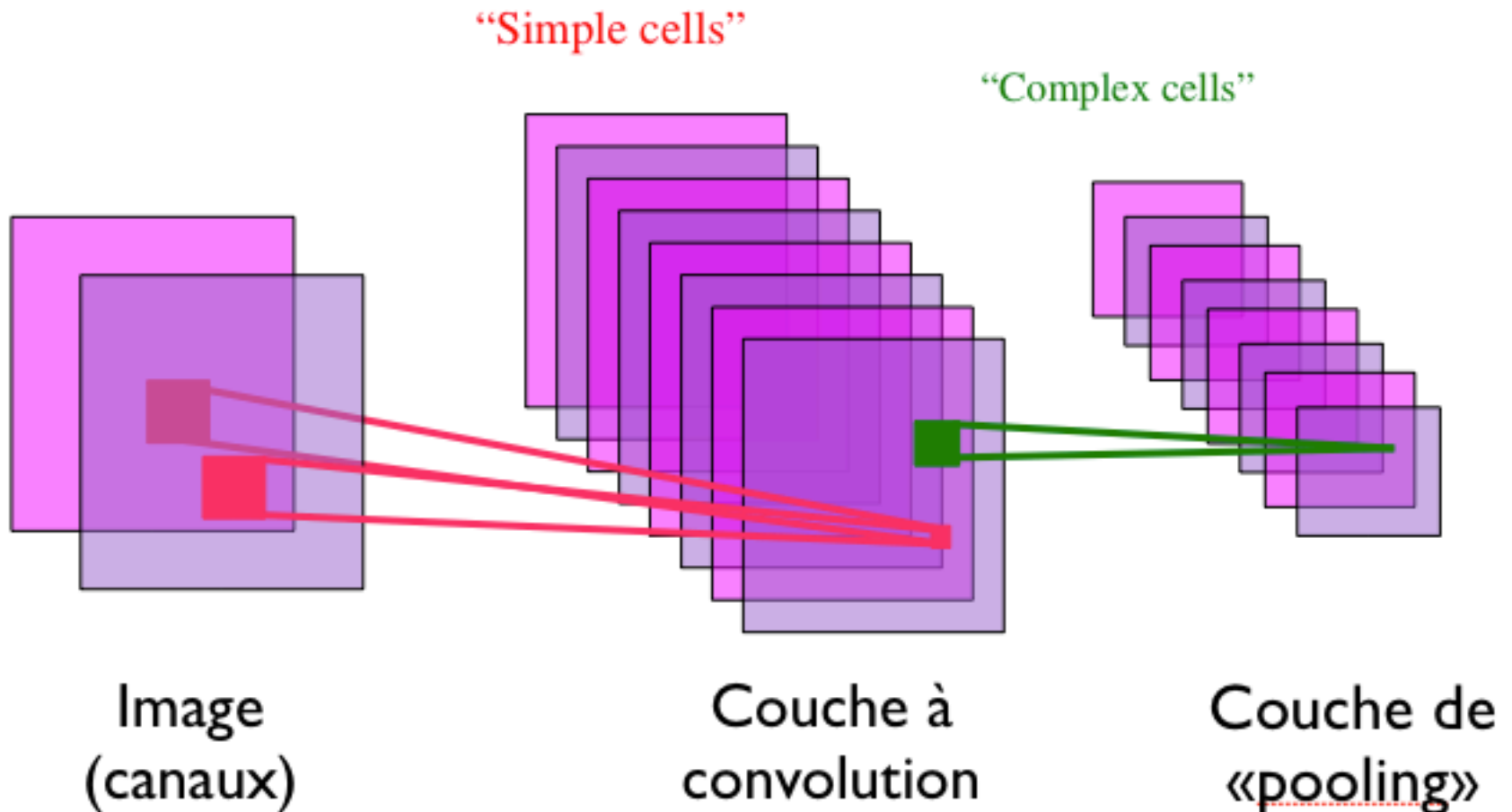


[picture from Simon Thorpe]

# Réseau de neurones à convolution

- Un **réseau de neurones à convolution** est cas spécial de réseau de neurones
  - ◆ Neocognition (Fukushima, 1980)
  - ◆ LeNet (LeCun, 1989)
- Comme un réseau de neurones standard, on l'entraîne par descente de gradient stochastique à l'aide de la rétropropagation des gradients
- Spécificité: ils implémentent **3 idées**:
  - ◆ connectivité parcimonieuse («sparse»)
  - ◆ connectivité locale
  - ◆ partage de paramètres

# Réseau de neurones à convolution: structure des couches cachées

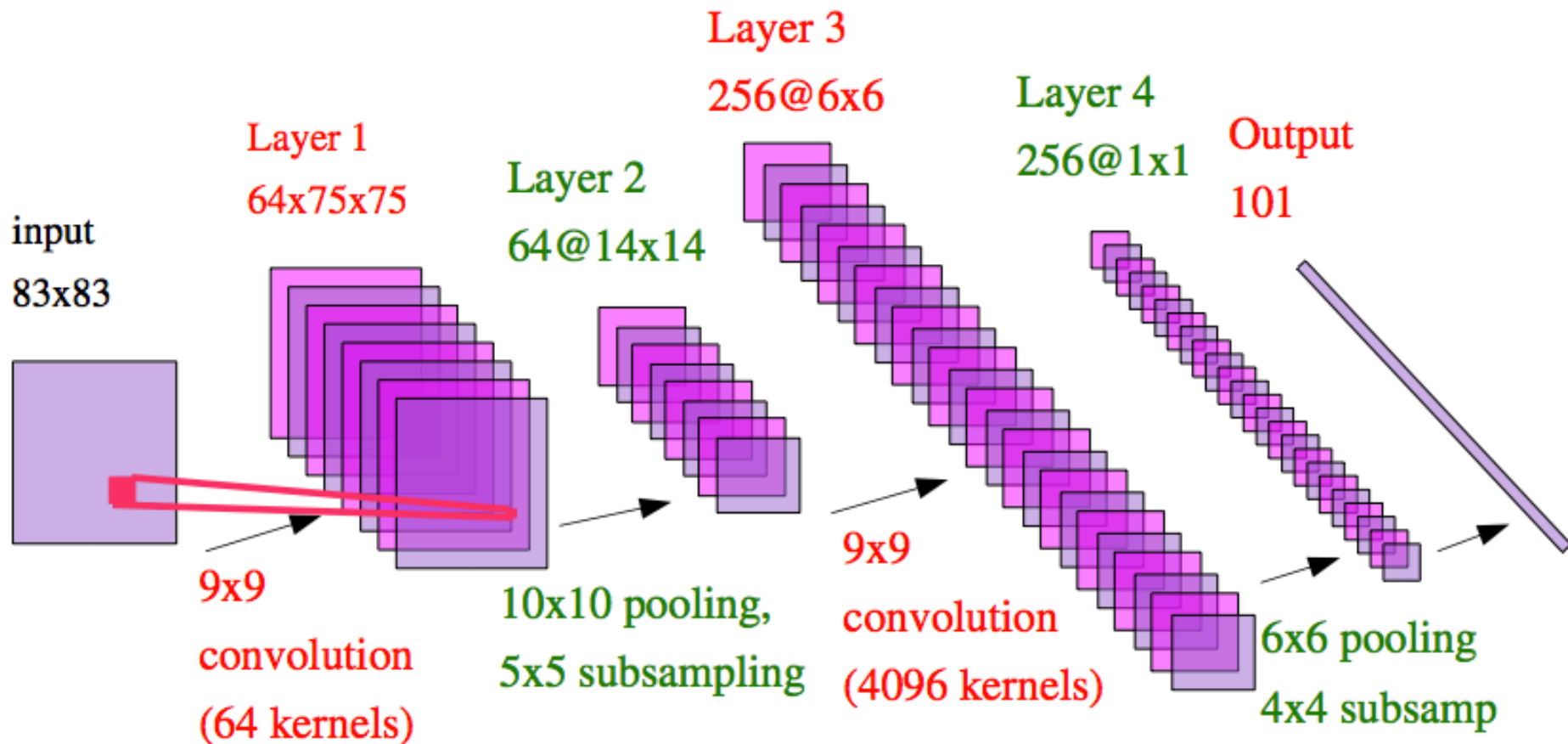


# Inspiration de la neuroscience

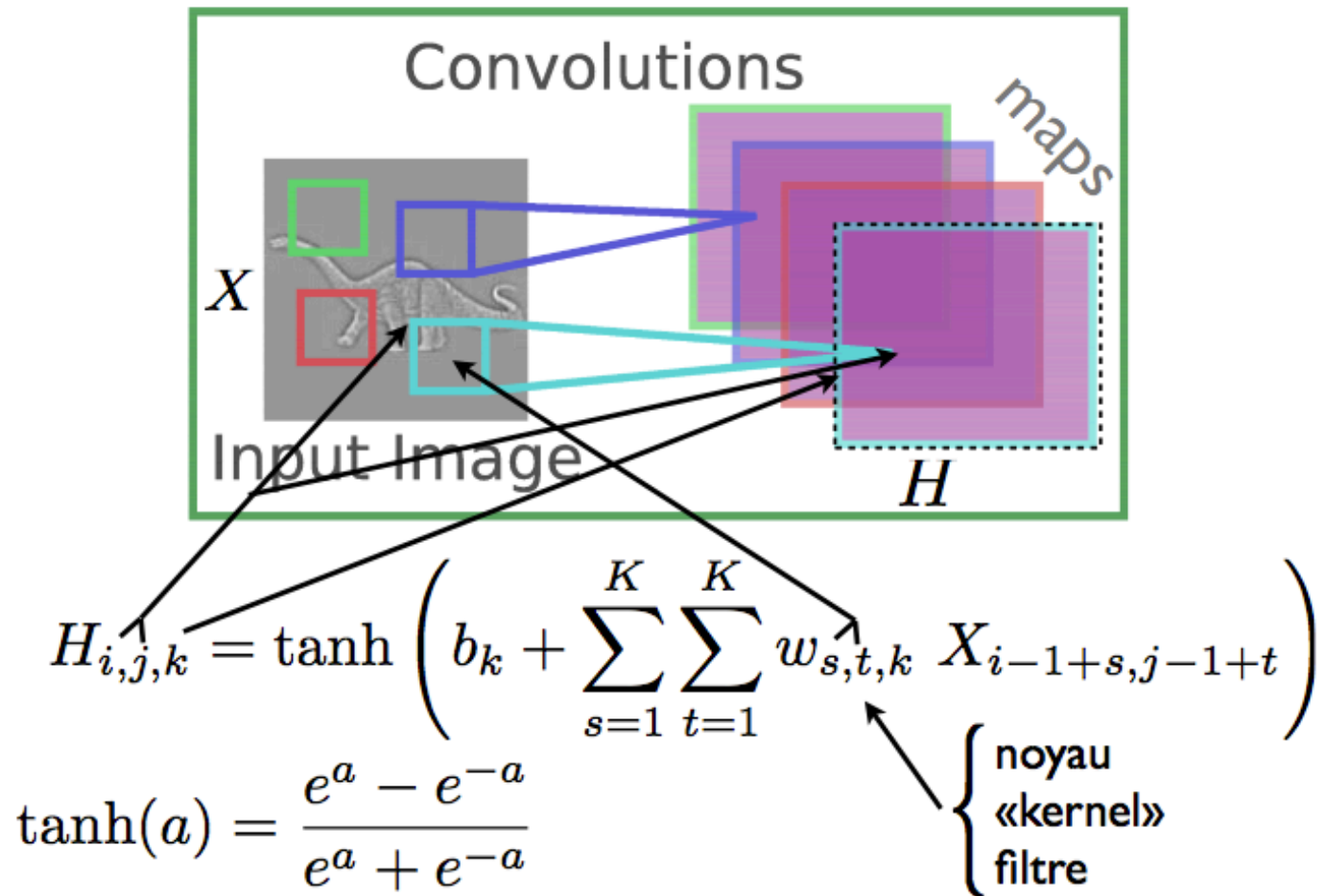
Hubel & Wiesel video

[http://www.youtube.com/watch?  
v=8VdFf3egwfg&feature=related](http://www.youtube.com/watch?v=8VdFf3egwfg&feature=related)

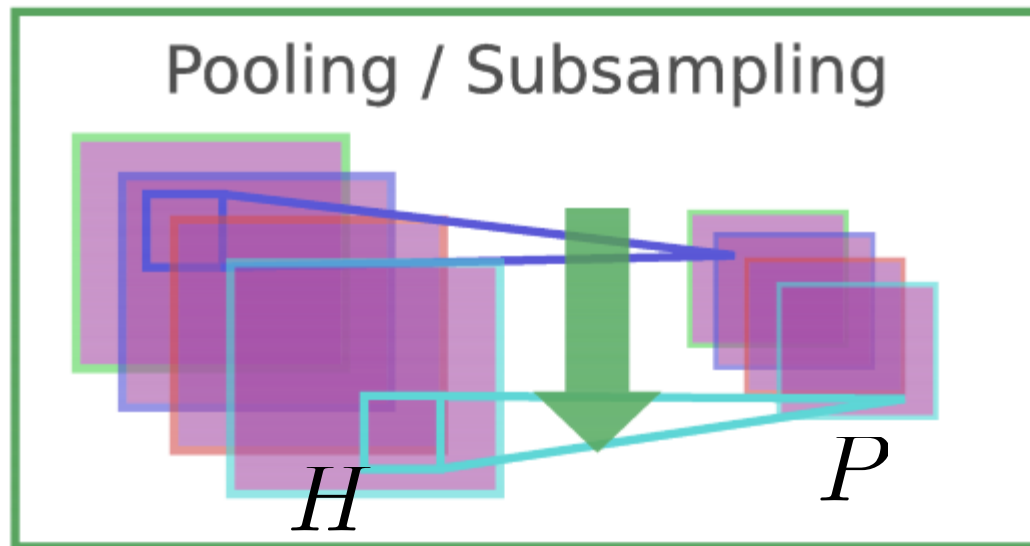
# Réseau de neurones à convolution: réseau complet



# Réseau de neurones à convolution: couche à convolution



# Réseau de neurones à convolution: couche à « pooling »



- Deux étapes

- ◆ «max pooling»: 
$$P_{i,j,k} = \max_{(i',j') \in N(i,j)} H_{i',j',k}$$
- ◆ «downsampling»: garde seulement une fraction des neurones

# Application: conduite automatique d'une voiture téléguidée



<http://www.cs.nyu.edu/~yann/research/dave/index.html>



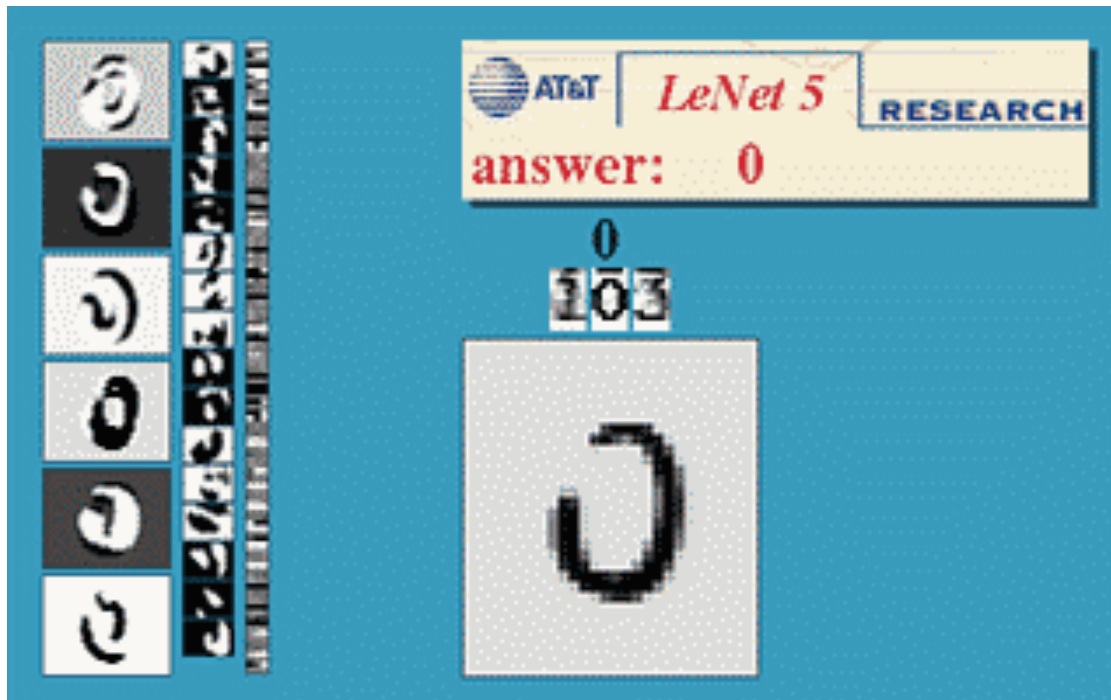
# Application: conduite automatique d'une voiture téléguidée



<http://www.cs.nyu.edu/~yann/research/dave/index.html>

# Application: et plusieurs autres

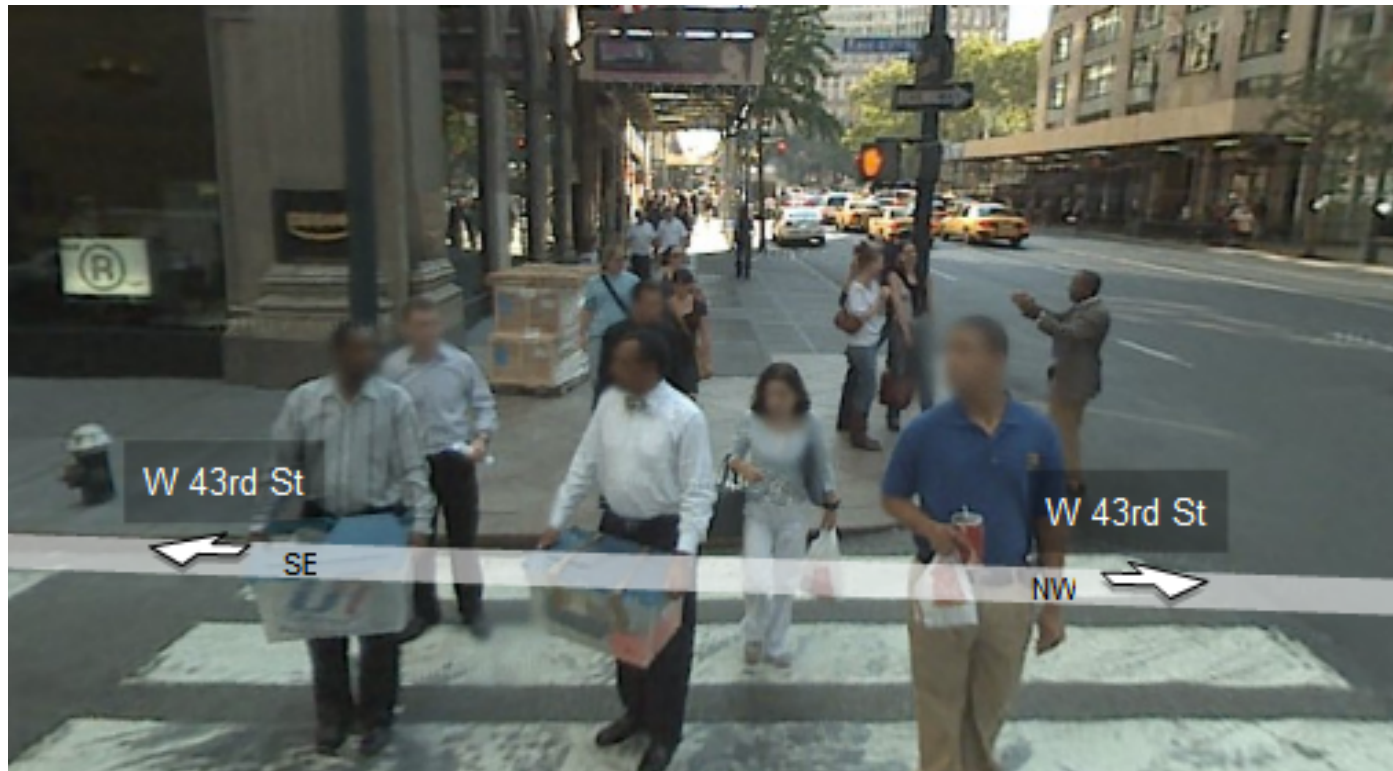
- Reconnaissance de caractères



<http://yann.lecun.com/exdb/lenet/>

# Application: et plusieurs autres

- Détection de visages et plaques d'immatriculation
  - ◆ utilisé dans Google Streetview pour masquer les visages automatiquement



# En rafale: autres concepts en vision par ordinateur

- Extraction de caractéristiques pour des images de couleur
  - ◆ en plus des histogrammes de gradients, on peut extraire des histogrammes des couleurs
    - » on détermine des intervalles de valeur pour R, G et B individuellement (ex.: 4 intervalles pour chaque canal)
    - » on considère chaque combinaison d'intervalle R, G et B comme une case (ex.:  $4^3 = 64$  cases)

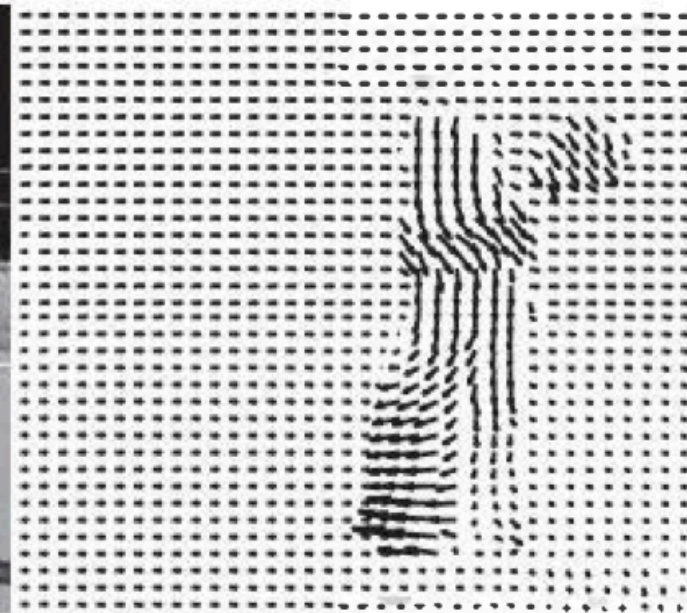
# En rafale: autres concepts en vision par ordinateur

- Extraction de caractéristiques pour la vidéo
  - ◆ on peut estimer le mouvement dans une image à l'aide du **flot optique**
  - ◆ à chaque pixel d'une image (*frame*), trouver le pixel dans l'image suivante qui est le plus « similaire » (c.-à-d. entouré de pixels similaires)



image  $t$

image  $t+1$



flot optique

# Conclusion

- Les technologies de vision par ordinateur sont de plus en plus performantes et de plus en plus répandues
  - ◆ détection de visage
  - ◆ détection de mouvements (*Microsoft Kinect*)
- Comme pour le traitement automatique de la langue, l'apprentissage automatique est de plus en plus au centre des technologies de vision par ordinateur
- Ce cours ne donne qu'une vue globale de la vision par ordinateur
  - ◆ le bacc en imagerie offre plusieurs cours sur le sujet (ex.: **IMN 559 - Vision par ordinateur**)
  - ◆ ces cours peuvent être suivis à la maîtrise...

# Vous devriez être capable de...

- Calculer une convolution
- Décrire globalement ce qu'est un contour et comment on peut les détecter
- Décrire ce qu'est un gradient d'image et connaître ses propriétés (norme vs. orientation)
- Décrire comment on extrait des caractéristiques d'une image à partir de ses gradients
- Savoir ce qui distingue un réseau de neurones à convolution d'autres sortes de réseaux de neurones