

# Vision par ordinateur - Partie 1

Cours 14, IFT 6758

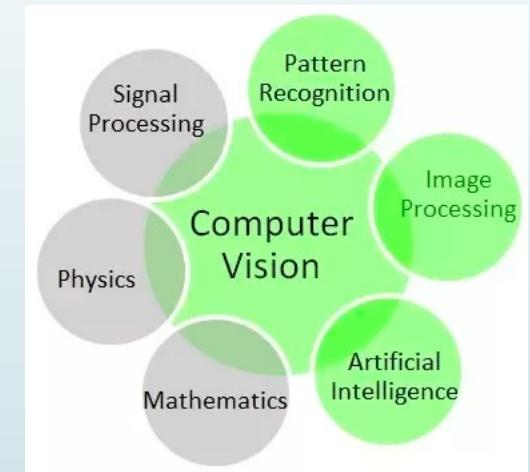
Automne 2024

# Aperçu

- ▶ Tâches en CV (vision par ordinateur)
- ▶ Les défis du CV
- ▶ Le pipeline CV
- ▶ Image d'entrée:
  - ▶ Prétraitement (la prochaine fois)
  - ▶ Extraction de caractéristiques (la prochaine fois)
  - ▶ Prédiction (la prochaine fois)

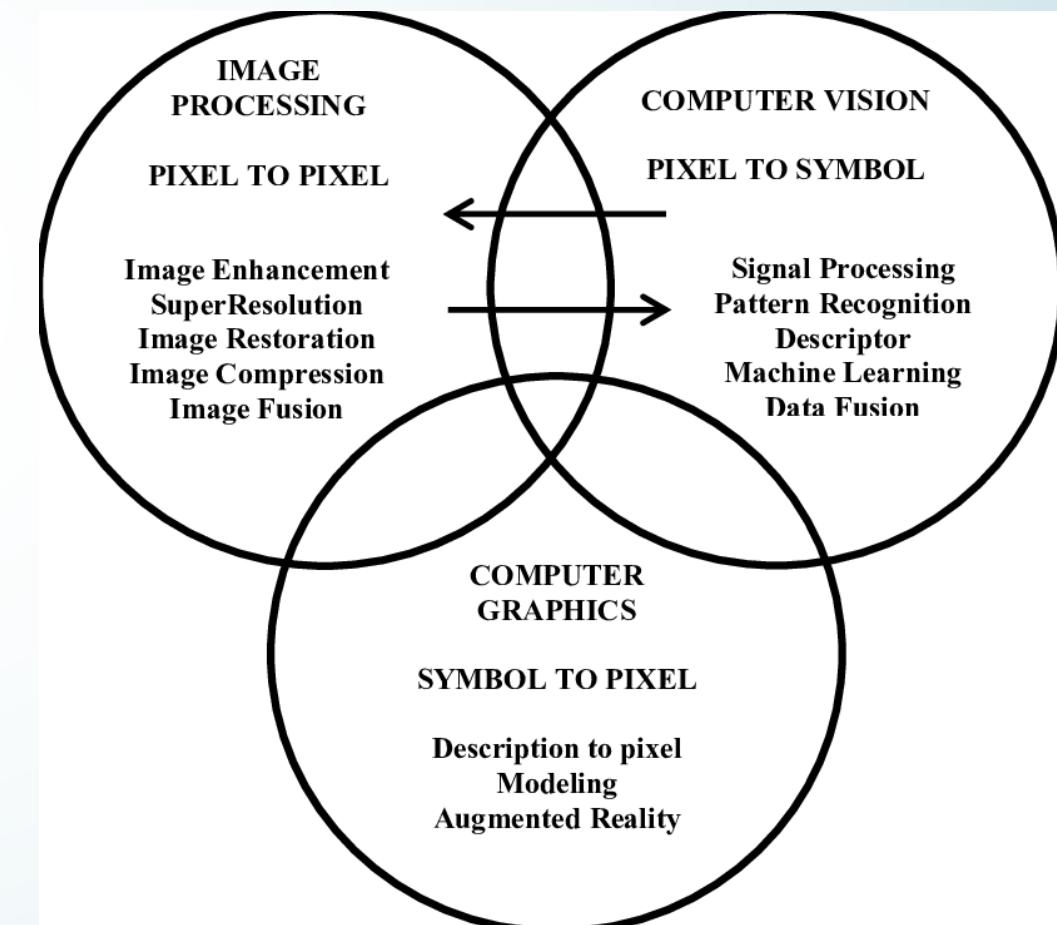
# Qu'est-ce que la vision par ordinateur?

- ▶ *La vision est l'acte de savoir ce qui est où en regardant. –Aristote*
- ▶ La vision par ordinateur est un domaine d'étude axé sur le problème **d'aider** les ordinateurs **à 'voir'**.



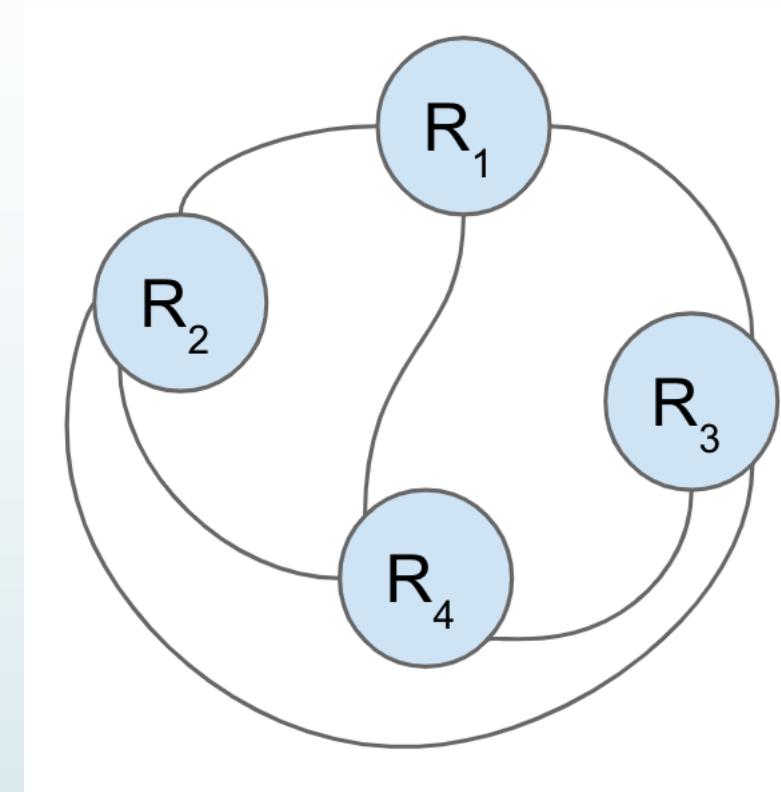
# Différence entre la vision par ordinateur et le traitement d'image

- ▶ La vision par ordinateur est **distincte** du traitement d'image.
- ▶ Le **traitement d'image** est le processus de **création d'une nouvelle image** à partir d'une image existante, simplifiant ou améliorant généralement le contenu d'une manière ou d'une autre.
- ▶ La **vision par ordinateur** s'intéresse à la **compréhension du contenu** d'une image.



# Tâches de vision

- ▶ Reconstruction
- ▶ Détection
- ▶ Réorganisation
- ▶ Reconnaissance



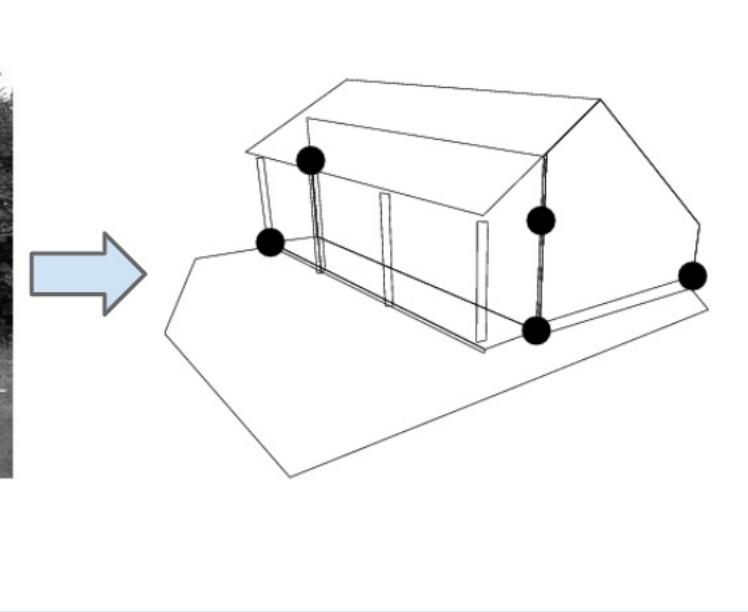
# Tâches de vision

- ▶ 1. La reconstruction



Left View

Right View



- ▶ Multiview Geometry, 3D Vision, Shape-from-X

# Tâches de vision

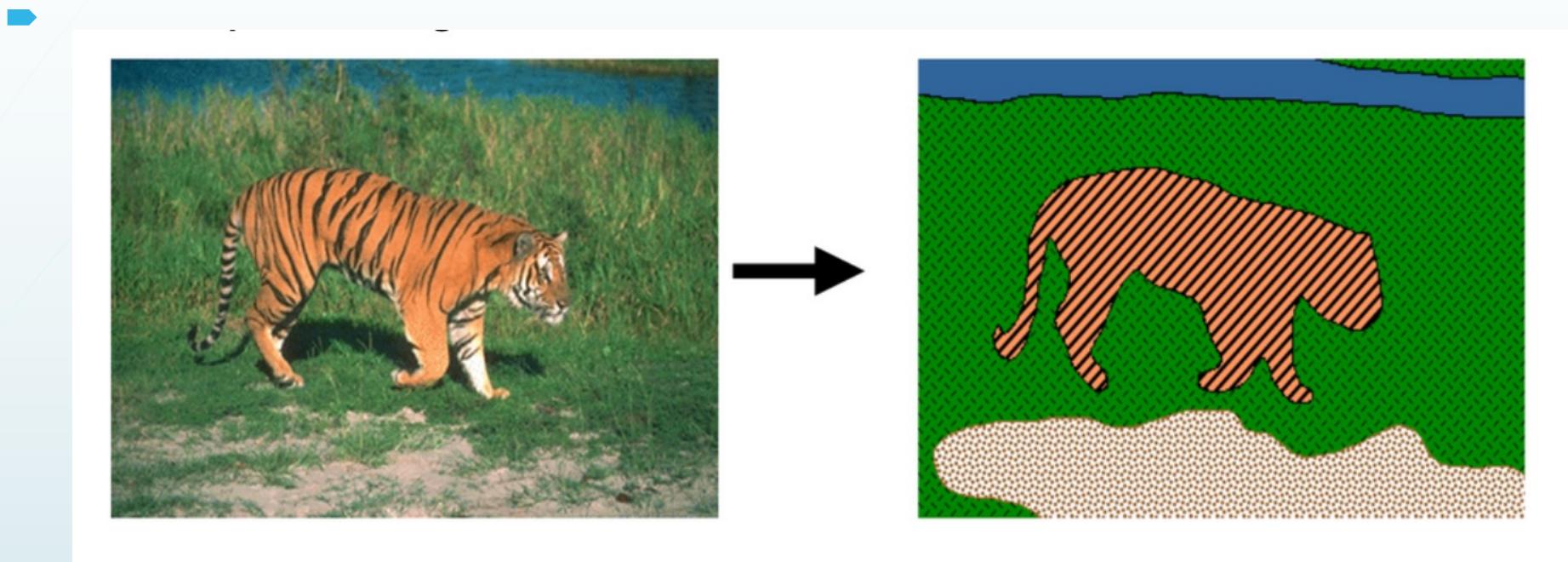
- ▶ 2. Détection



- ▶ Suivi, alignement, flux optique, correspondance

# Tâches de vision

- ▶ 3. Réorganisation



- ▶ Clustering, apprentissage non supervisé, segmentation, organisation perceptuelle

# Tâches de vision

## ► 4. Reconnaissance



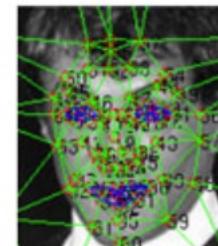
[Ricoh]



(a)



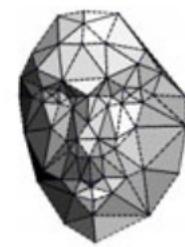
(b)



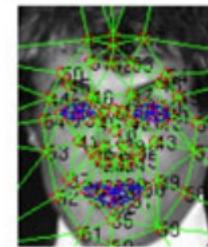
(c)



(d)



(e)



(f)



(g)



(h)

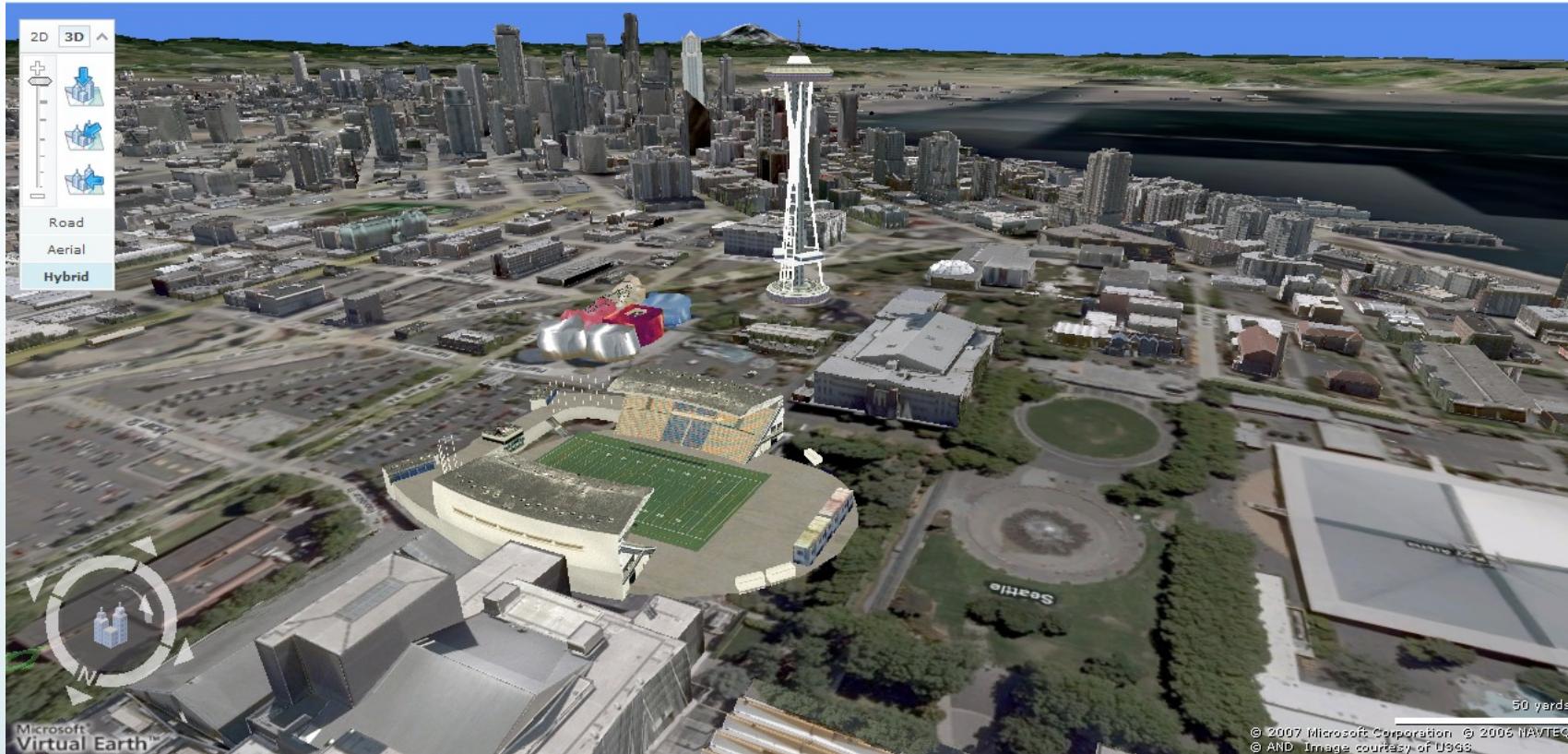
[DeepFace]

## ► Vérification, identification, détection

# Pourquoi étudier la vision par ordinateur?

- ▶ Les images et les films sont partout
- ▶ Collection croissante d'applications utiles
  - ▶ construire des représentations du monde 3D à partir d'images
  - ▶ Surveillance automatisée (qui fait quoi)
  - ▶ Post-traitement vidéo
  - ▶ Recherche de visage
- ▶ Une meilleure compréhension de la vision humaine

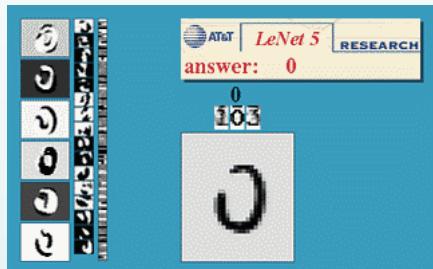
# Vue de la Terre (modélisation 3D)



- ▶ Image de Virtual Earth de Microsoft (voir aussi : Google Earth)

# Reconnaissance optique de caractères (OCR)

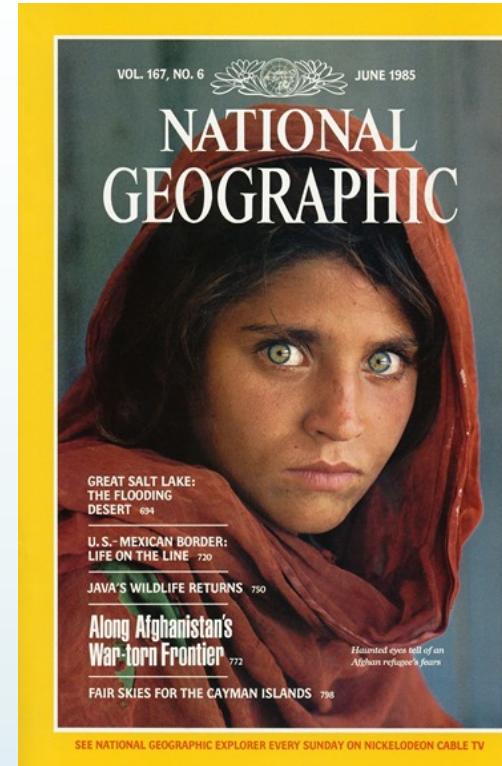
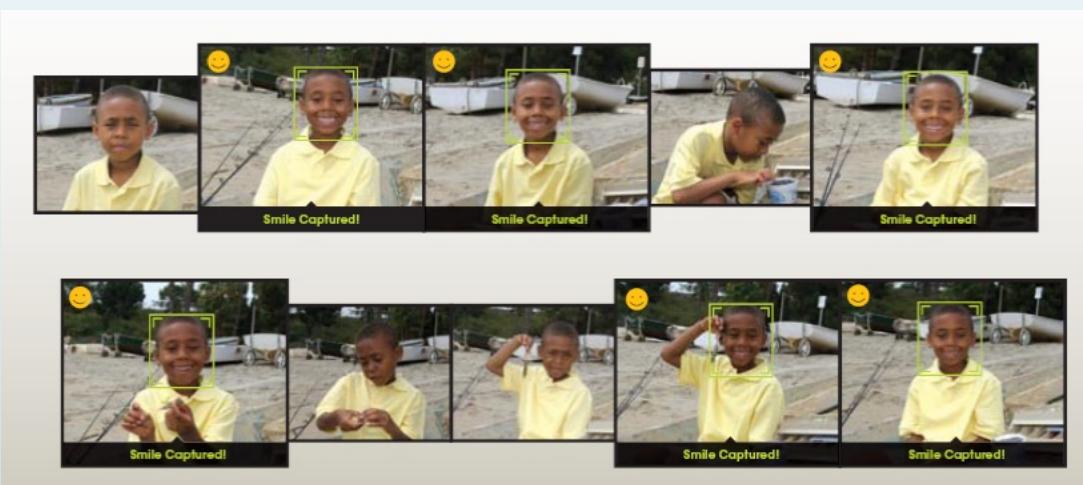
- ▶ Technologie pour convertir des documents numérisés en texte
  - ▶ Si vous avez un scanner, il est probablement livré avec un logiciel OCR



- ▶ Reconnaissance de chiffres, laboratoires AT&T
- ▶ Lecteurs de plaques d'immatriculation

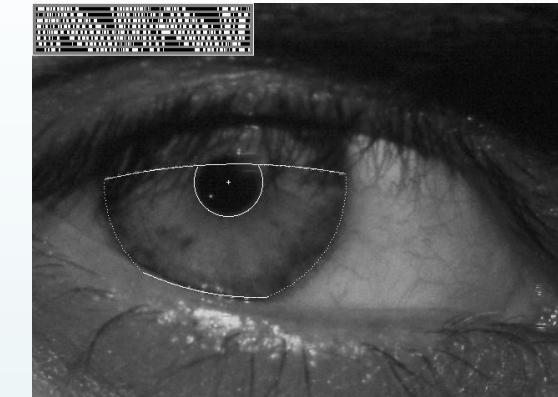
# Détection du visage et du sourire

- ▶ De nombreux nouveaux appareils photo numériques détectent désormais les visages
  - ▶ Canon, Sony, Fuji, ...
  - ▶ Sony Cyber-shot T70 appareil photo numérique



# Biométrie de la vision

- ▶ “How the Afghan Girl was Identified by Her Iris Patterns” lire l’histoire: [[ici](#)]



Scanners d'empreintes digitales sur de nombreux nouveaux ordinateurs portables,  
Autres appareils

<https://sensiblevision.com/>

# Reconnaissance d'objets

► [LaneHawk](#)



► [CamFind](#)



# Sports et jeux

- ▶ Sportvision first down line; [explanation](#)



- ▶ [Créer un avatar 3D à partir d'une image](#)



- ▶ [Nintendo Wii](#) dispose d'un système de suivi basé sur une caméra infrarouge



# Robotique

► [NASA Mars Spirit Rover](#)

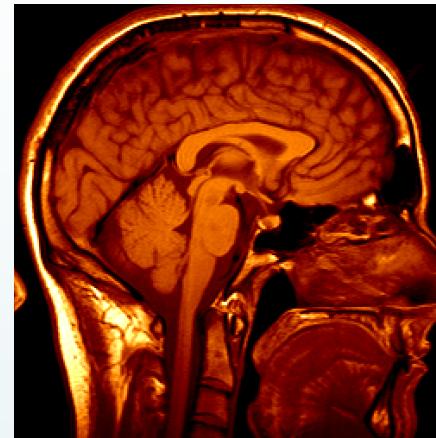


► [Robocup](#)



# Imagerie médicale

► Imagerie 3D : IRM, TDM



► Chirurgie guidée par l'image

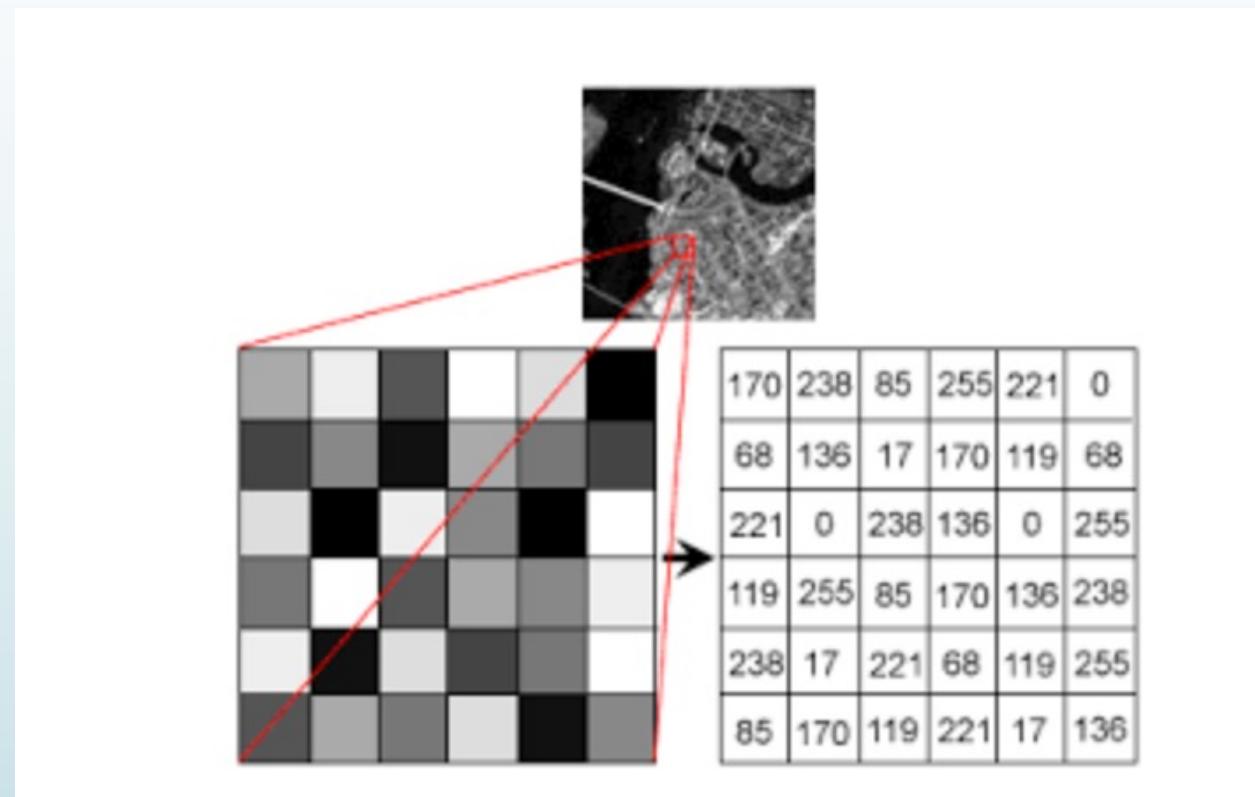




## Défis de la vision par ordinateur

# Comment les machines voient-elles une image ?

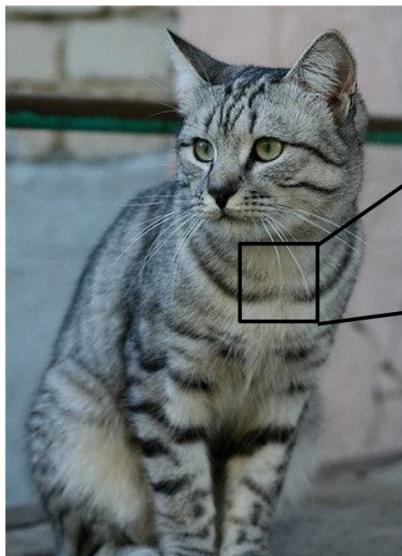
- ▶ Les machines voient et traitent tout à l'aide de chiffres, y compris des images et du texte. Comment convertir des images en nombres ?



# Image

- Chaque nombre représente **l'intensité** à cet endroit particulier. Par exemple, pour une image en niveaux de gris où chaque pixel ne contient qu'une seule valeur, c'est-à-dire l'intensité de la couleur noire à cet endroit.

## The Problem: Semantic Gap



This image by Nikita is  
licensed under CC-BY 2.0

```
[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]
[ 91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]
[ 76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85]
[ 98 81 81 93 120 131 127 108 95 98 102 99 96 93 101 94]
[106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95]
[133 137 147 183 65 81 80 65 52 54 74 84 102 93 85 82]
[114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
[123 137 147 183 65 81 80 65 52 54 74 84 102 93 85 82]
[123 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98]
[127 125 131 147 133 127 126 131 111 98 89 75 61 64 72 84]
[115 114 109 123 150 148 131 118 113 108 109 92 74 65 72 78]
[ 89 93 90 97 108 147 131 118 113 114 113 108 106 95 77 80]
[ 63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]
[ 62 65 82 89 78 71 88 101 124 126 119 101 107 114 131 119]
[ 63 65 75 88 89 71 62 81 120 138 135 185 81 98 110 118]
[ 87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112]
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107]
[164 146 112 80 82 120 124 104 76 48 45 66 88 101 102 109]
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]
[130 128 134 161 139 100 109 118 121 134 114 87 65 53 69 86]
[128 112 96 117 150 144 128 115 104 107 102 93 87 81 72 79]
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]
[122 121 182 80 82 86 94 117 145 148 153 182 58 78 92 107]
[122 164 148 103 71 56 78 83 93 103 119 139 182 61 69 84]]
```

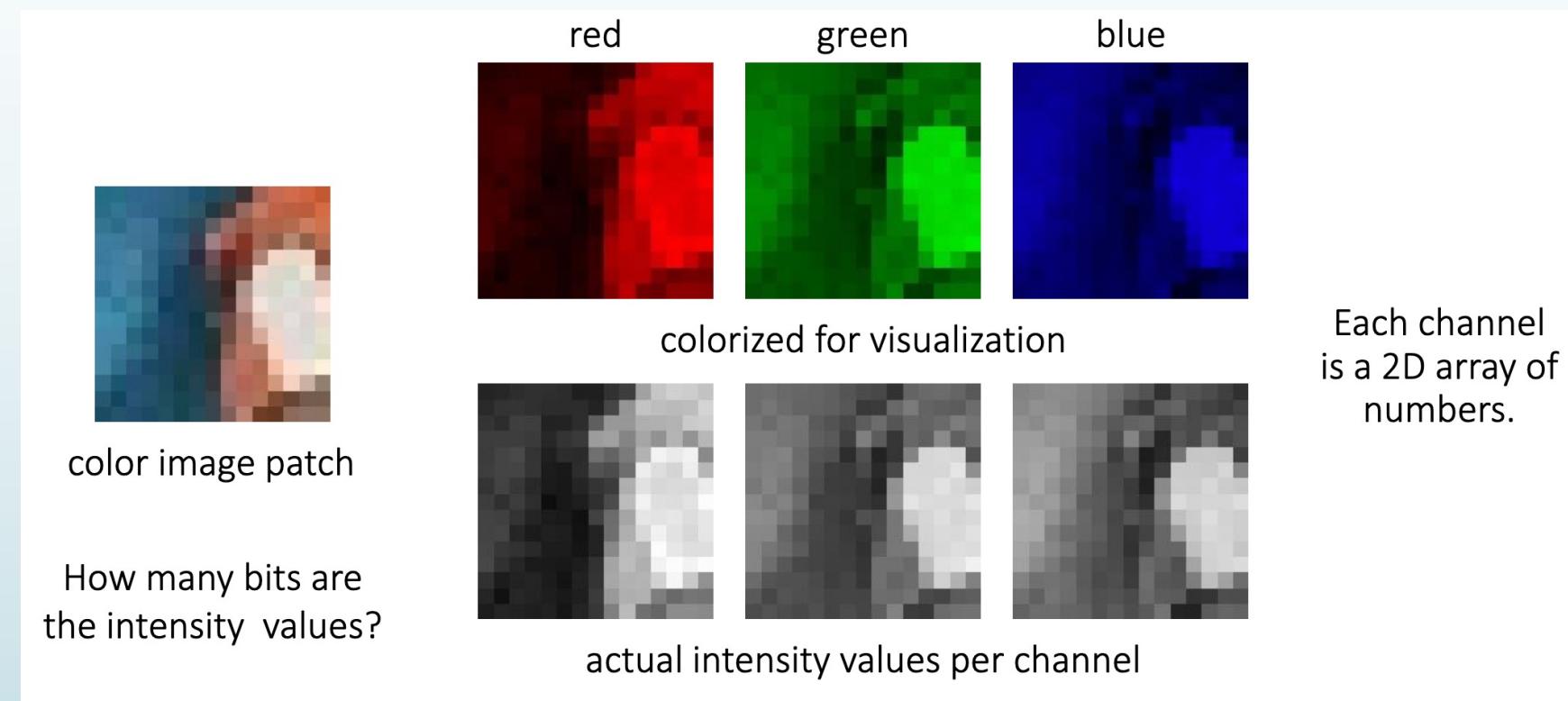
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3  
(3 channels RGB)

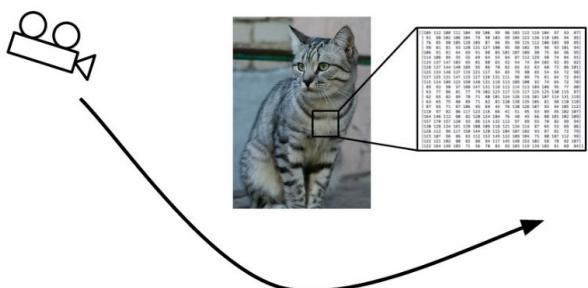
# Qu'est-ce qu'une image ?

- ▶ Les images couleur auront plusieurs valeurs pour un seul pixel. Ces valeurs représentent **l'intensité des canaux respectifs – canaux rouge, vert et bleu** pour les images RVB, par exemple.



# Les défis de la reconnaissance

**Viewpoint**



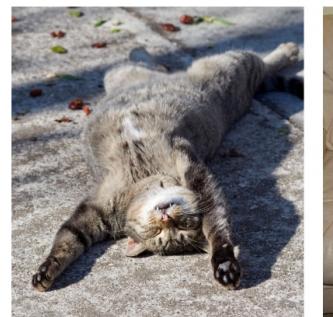
This image is CC0 1.0 public domain

**Illumination**



This image by Umberto Salvagnin is licensed under CC-BY 2.0

**Deformation**

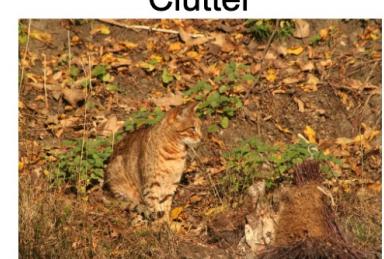


This image by jonsson is licensed under CC-BY 2.0

**Occlusion**



**Clutter**



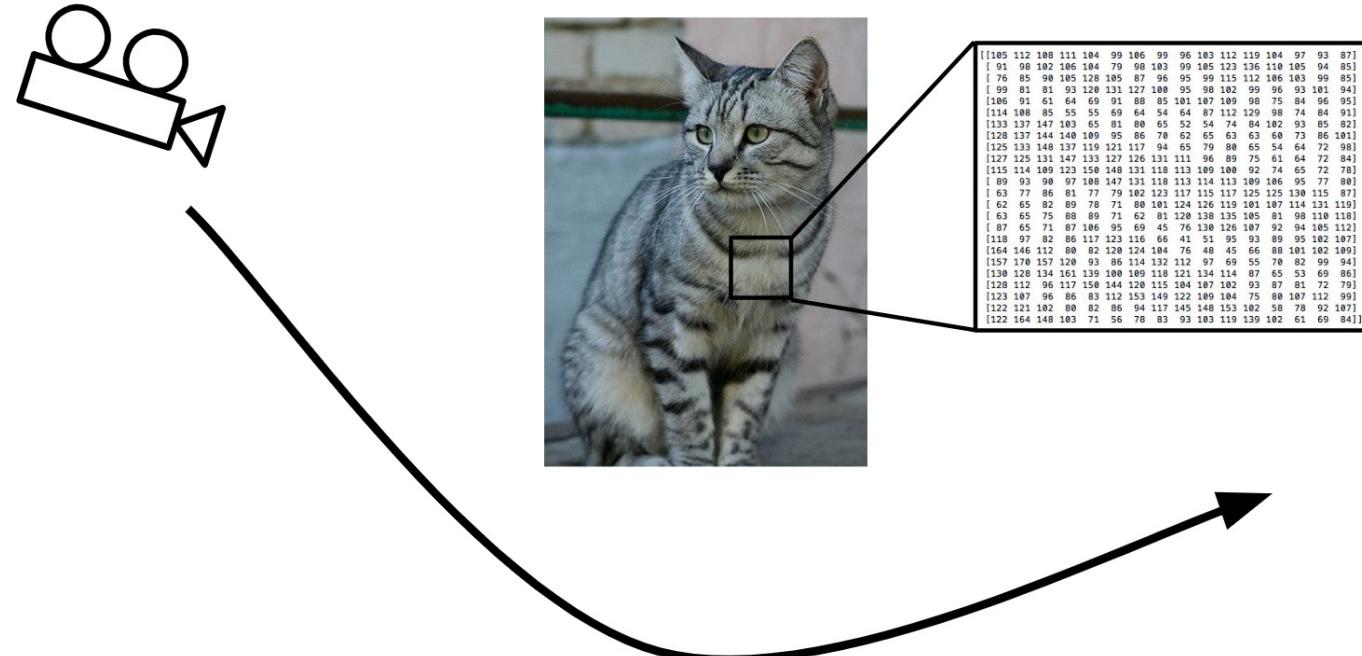
This image is CC0 1.0 public domain

**Intraclass Variation**



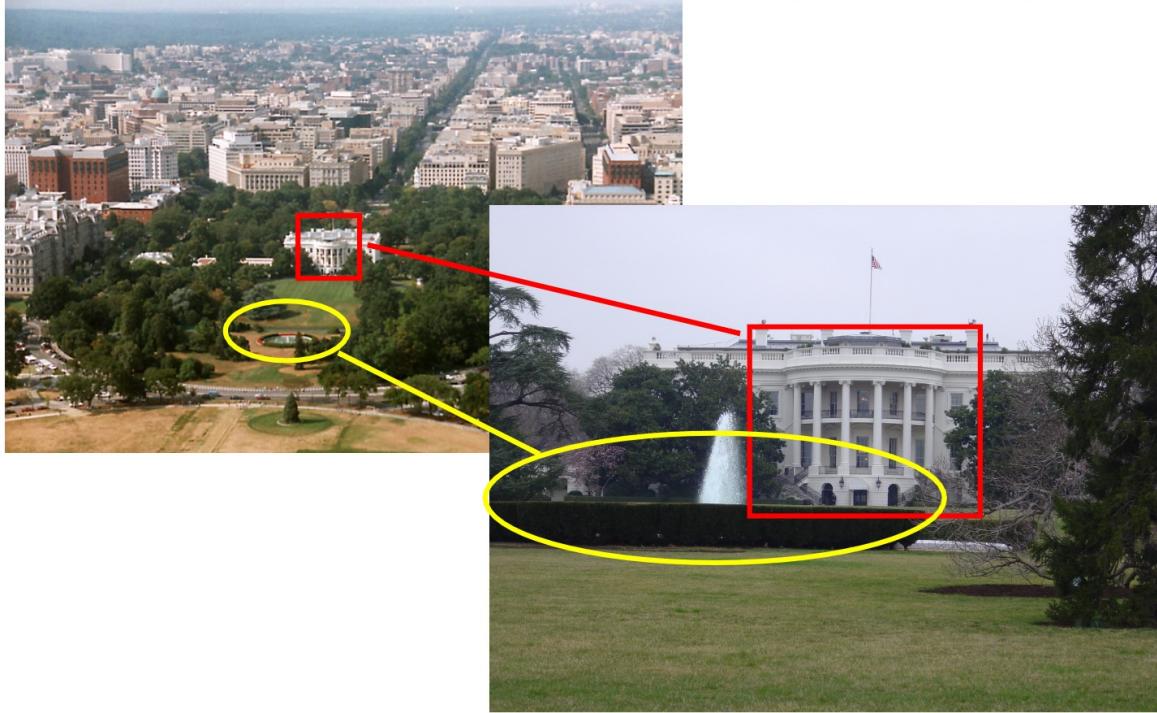
This image is CC0 1.0 public domain

# Défi : Variation des points de vue



# Défi : Variation des points de vue

**Object appearance changes with respect to viewpoint**



# Défi : Illumination

L'apparence de l'objet change en fonction de l'amplitude et de la direction de l'éclairage.



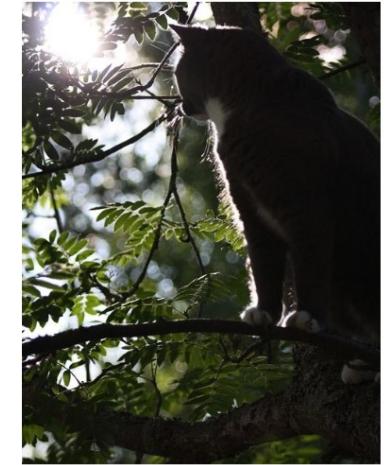
[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

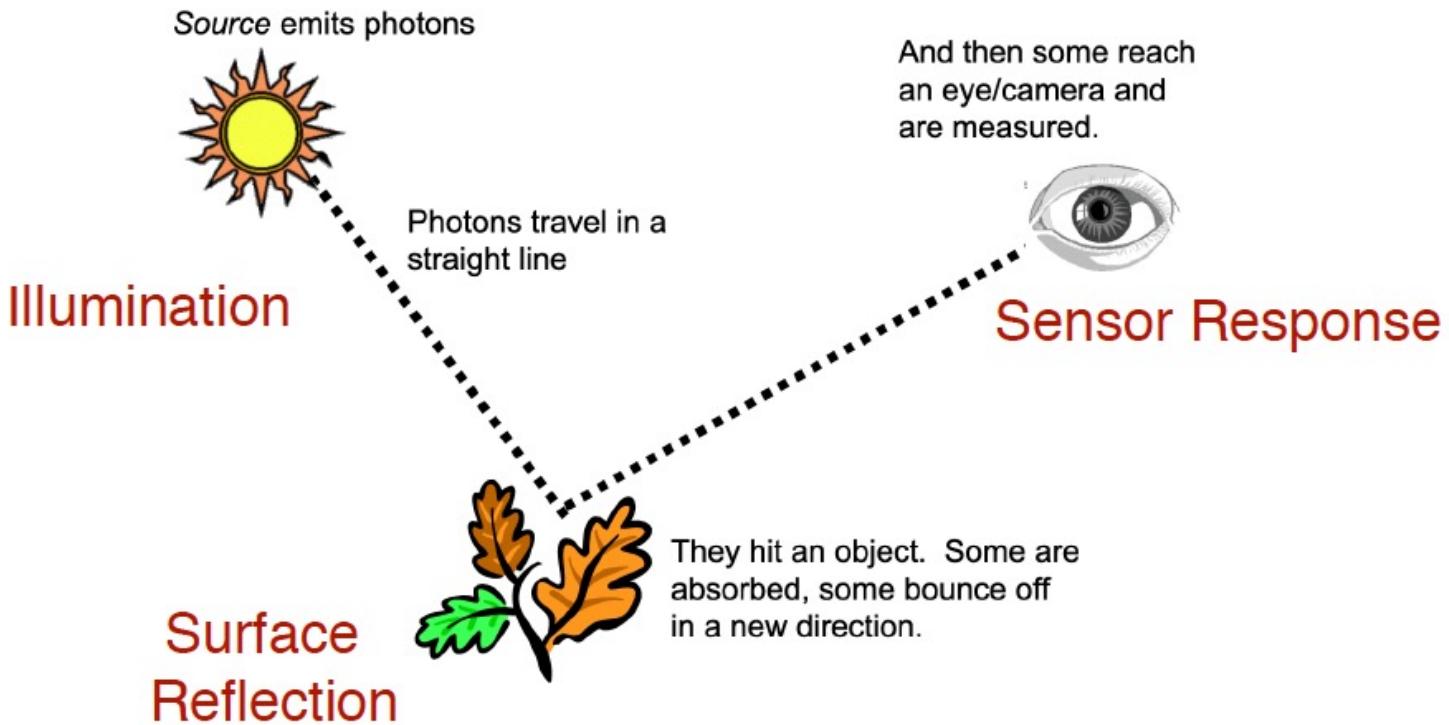


[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

# Qu'est-ce que la couleur ?



# Couleur

- ▶ La perception des couleurs est une composition de trois facteurs (**éclairage, réflectance de surface, réponse du capteur**)
- ▶ Nous ne pouvons pas facilement utiliser la couleur que nous voyons dans l'image pour déduire l'éclairage et le matériau (même si les propriétés du capteur sont fixes et connues).
- ▶ “rouge” signifie “apparaît comme rouge pour un humain lorsque la lumière ambiante est blanche”
- ▶ Un objet blanc apparaîtra comme rouge si la lumière est rouge
- ▶ Les daltoninens ne distinguent pas certaines couleurs



“normal” color perception



red/green color blind

## La robe est-elle bleue et noire ou blanche et or?

- ▶ Cette robe parvient à rassembler simultanément plus de 670 000 personnes sur Buzzfeed et à convaincre 900 000 visiteurs de passer un sondage.



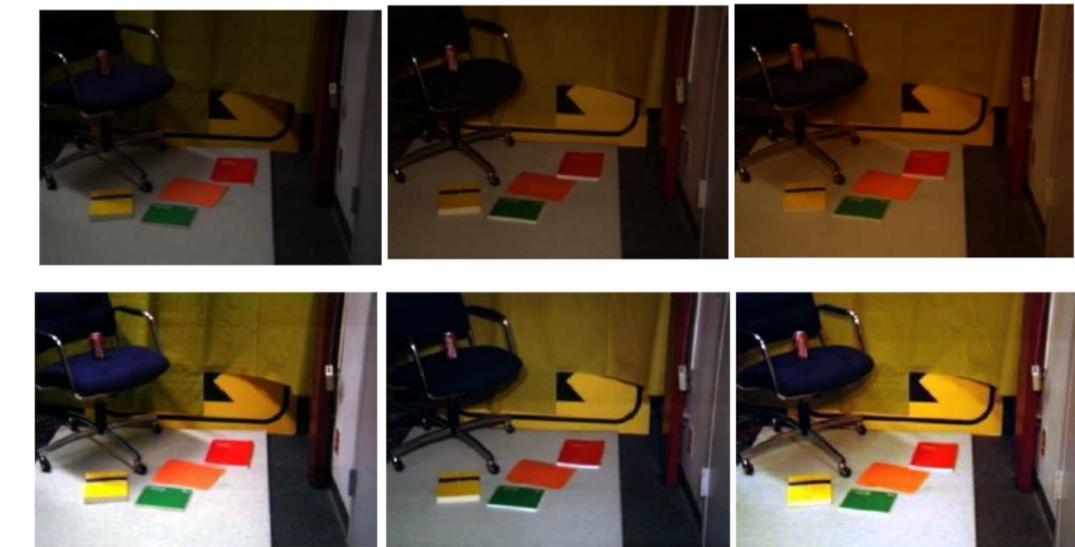
# La réponse est contextuelle



# Défi : Constance des couleurs

Les humains sont très fort à reconnaître la même ‘couleur’ sour différentes conditions

Humans are very good at recognizing the same material colors under different illumination. Not clear how this is achieved in the general case.



# Défi : Déformation



This image by Umberto Salvagnin  
is licensed under [CC-BY 2.0](#)



This image by Umberto Salvagnin  
is licensed under [CC-BY 2.0](#)



This image by sare bear is  
licensed under [CC-BY 2.0](#)



This image by Tom Thai is  
licensed under [CC-BY 2.0](#)

# Défi : Occlusion



[This image](#) is CC0 1.0 public domain



[This image](#) is CC0 1.0 public domain



[This image](#) by [jonsson](#) is licensed  
under CC-BY 2.0

# Défi : Variation



[This image is CC0 1.0 public domain](#)

# Mesure de distance pour comparer les images

**L1 distance:**

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

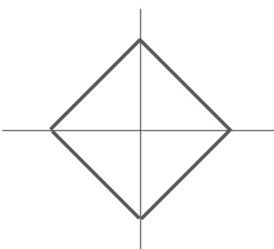
test image				training image				pixel-wise absolute value differences			
56	32	10	18	10	20	24	17	46	12	14	1
90	23	128	133	8	10	89	100	82	13	39	33
24	26	178	200	12	16	178	170	12	10	0	30
2	0	255	220	4	32	233	112	2	32	22	108

-

$\Rightarrow$  add 456

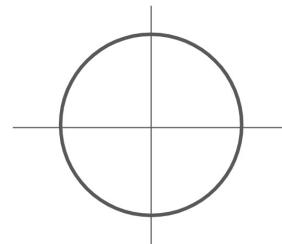
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

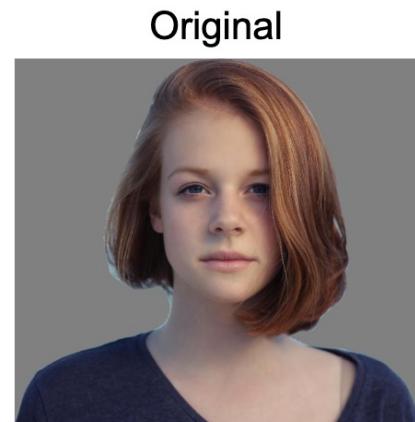


L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



# Mesures de distance sur les pixels



Original



Boxed



Shifted



Tinted

[Original image is  
CC0 public domain](#)

(all 3 images have same L2 distance to the one on the left)

Plus proche voisins n'est pas utilisé pour les images.

- Très lent
- Les distances dans l'espace des pixels sont très peu utiles

# Distance dans L'espace des pixels

Paire d'images (ligne 1 vs. ligne 2). Quelles sont les images les plus proches?

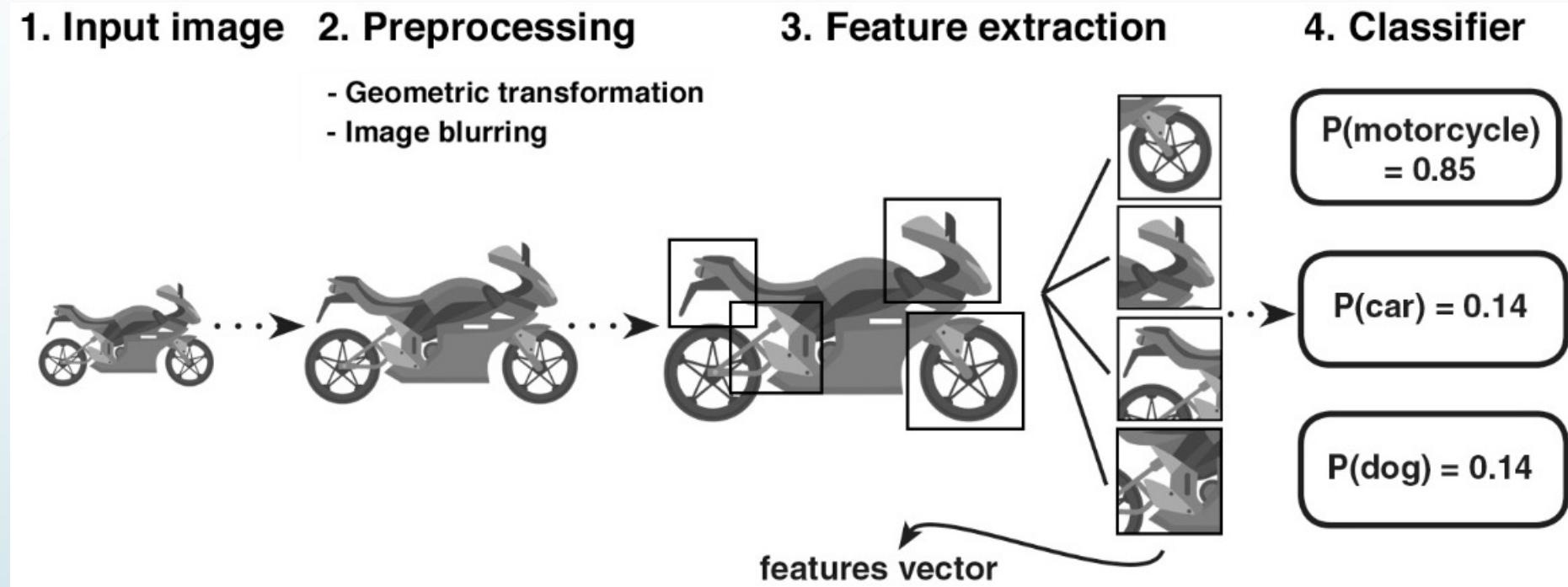




# Vision par ordinateur: principales opérations

# Pipeline de vision par ordinateur

1. Entrée. 2. Pre-traitement. 3 Extraction de caractéristiques. 4. prédiction



# Opérations principales de vision par ordinateur

## Filtering and Smoothing



Linear operators  
Convolution  
Smoothing

# Opérations principales de vision par ordinateur

## Feature Extraction

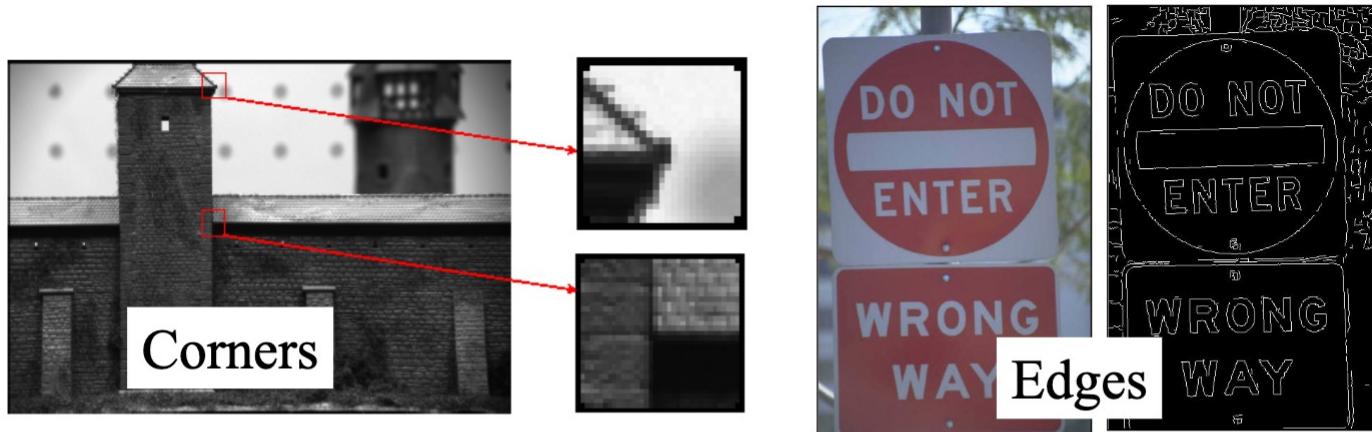


Image derivatives  
Gradient operators  
DoG/LoG operators  
Harris corner detector

**Why?**  
**Seek more unique descriptors  
(than pixels) for matching**

# Opérations principales de vision par ordinateur

## Color and Light

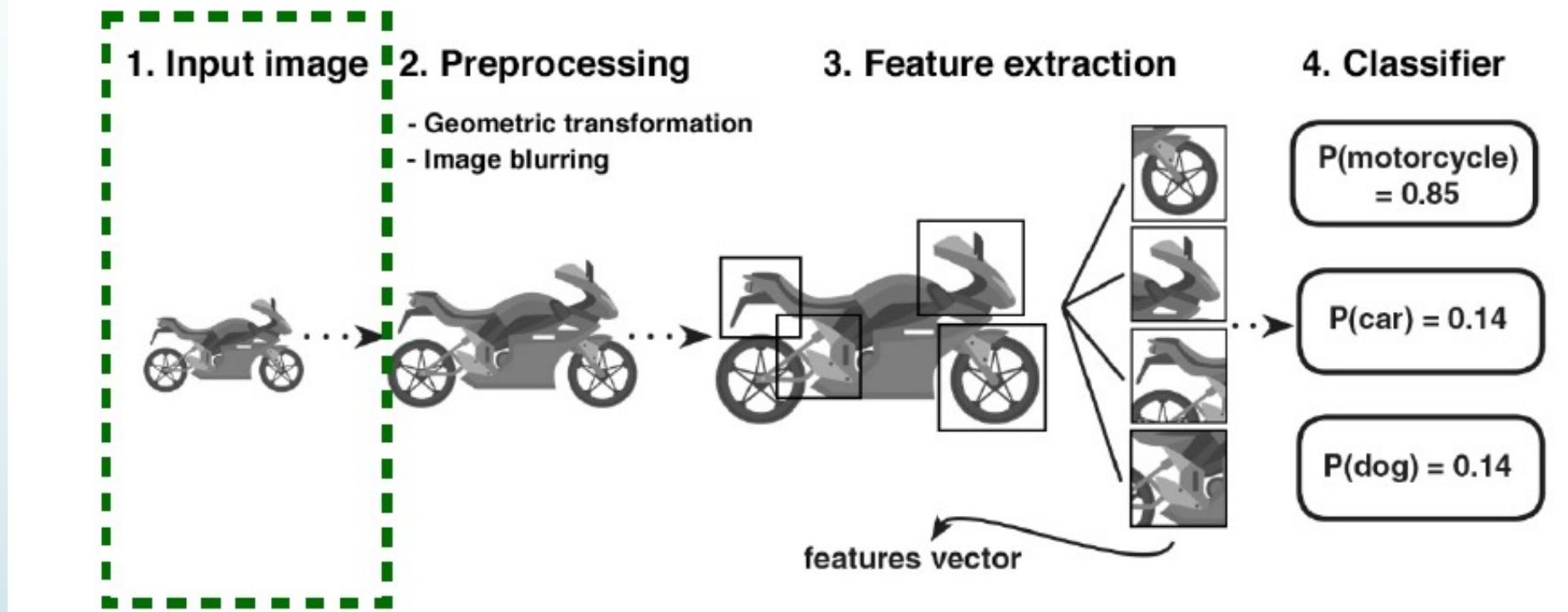


Radiance / Reflection  
Illumination / Shading  
Chromaticity  
Color Constancy



# Pipeline de vision par ordinateur

1. Entrée. 2. Pre-traitement. 3 Extraction de caractéristiques. 4. prédiction



# Images en tant que fonctions

- **Une image** est une fonction de  $R^2$  dans  $R^M$ 
  - $f(x, y)$  donne **l'intensité** de chaque canal à la position  $(x, y)$
  - Définie dans un rectangle (d'habitude  $[0,1]^2$ )
    - $f: [0,1] \times [0,1] \rightarrow [0,255]^3$

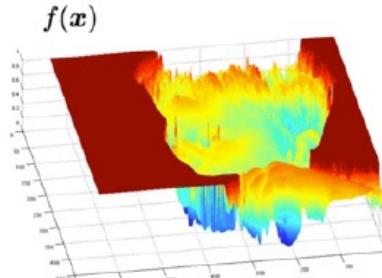
- **An Image as a function  $f$  from  $R^2$  to  $R^M$ :**
  - $f(x, y)$  gives the **intensity** at position  $(x, y)$
  - Defined over a rectangle, with a finite range:

$$f: \underbrace{[a,b] \times [c,d]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$



grayscale image

What is the range of  
the image function  $f$ ?



$$\text{domain } \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

A (grayscale)  
image is a 2D  
function.

# Images en tant que fonctions

- Une image est une fonction de  $R^2$  dans  $R^M$ 
  - $f(x, y)$  donne l'intensité de chaque canal à la position  $(x, y)$
  - Définie dans un rectangle (d'habitude  $[0,1]^2$ )
    - $f: [0,1] \times [0,1] \rightarrow [0,255]^3$

- **An Image as a function  $f$  from  $R^2$  to  $R^M$ :**
  - $f( x, y )$  gives the **intensity** at position  $( x, y )$
  - Defined over a rectangle, with a finite range:

$$f: \underbrace{[a,b] \times [c,d]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

- A color image:  $f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$

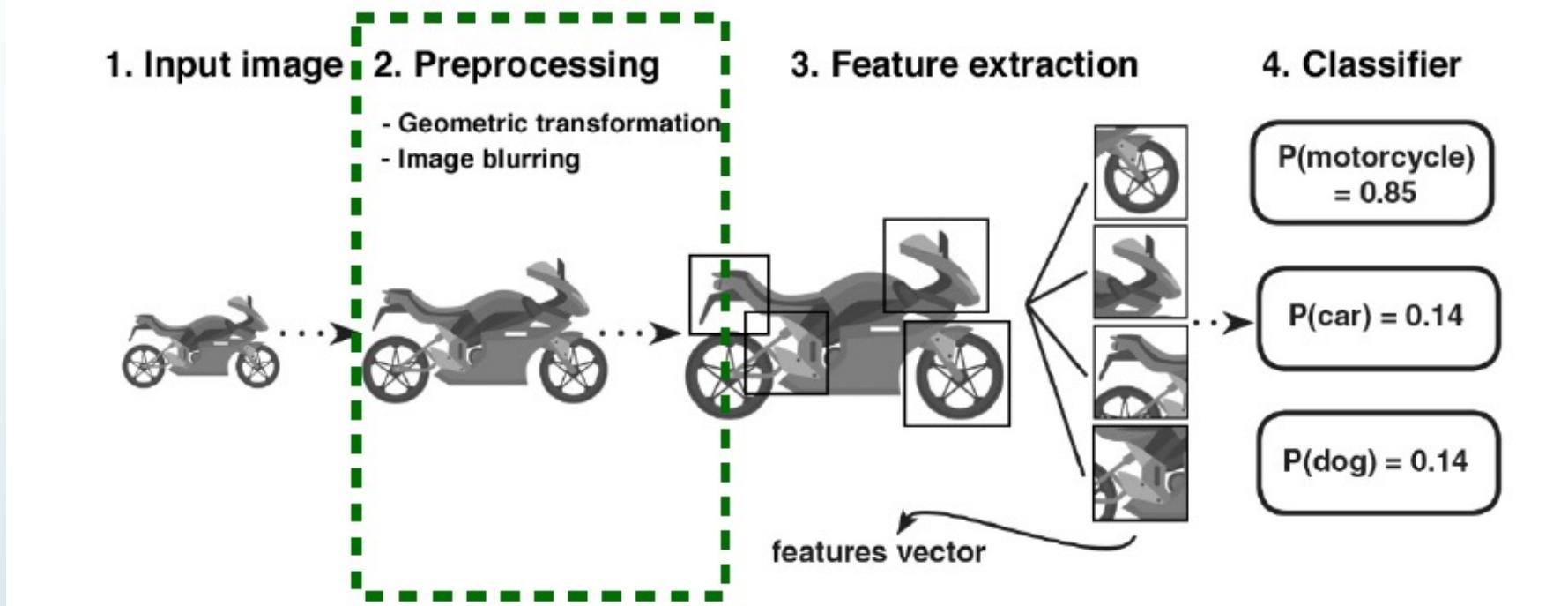
# Image d'entrée

- ▶ Par défaut, la fonction **imread** lit les images au format BGR (bleu-vert-rouge). Nous pouvons lire des images dans différents formats en utilisant des drapeaux supplémentaires dans la fonction imread:
- ▶ **CV2.IMREAD\_COLOR** : indicateur par défaut pour le chargement d'une image couleur
- ▶ **CV2.IMREAD\_GRAYSCALE** : charge les images au format niveaux de gris
- ▶ **CV2.IMREAD\_UNCHANGED** : charge les images dans leur format donné, y compris le canal alpha. Le canal alpha stocke les informations de transparence – plus la valeur du canal alpha est élevée, plus le pixel est opaque

```
1 #import the libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import cv2
5 %matplotlib inline
6
7 #reading the image
8
9 image = cv2.imread('index.png')
10 image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
11 #plotting the image
12 plt.imshow(image)
13
14 #saving image
15 cv2.imwrite('test_write.jpg',image)
```

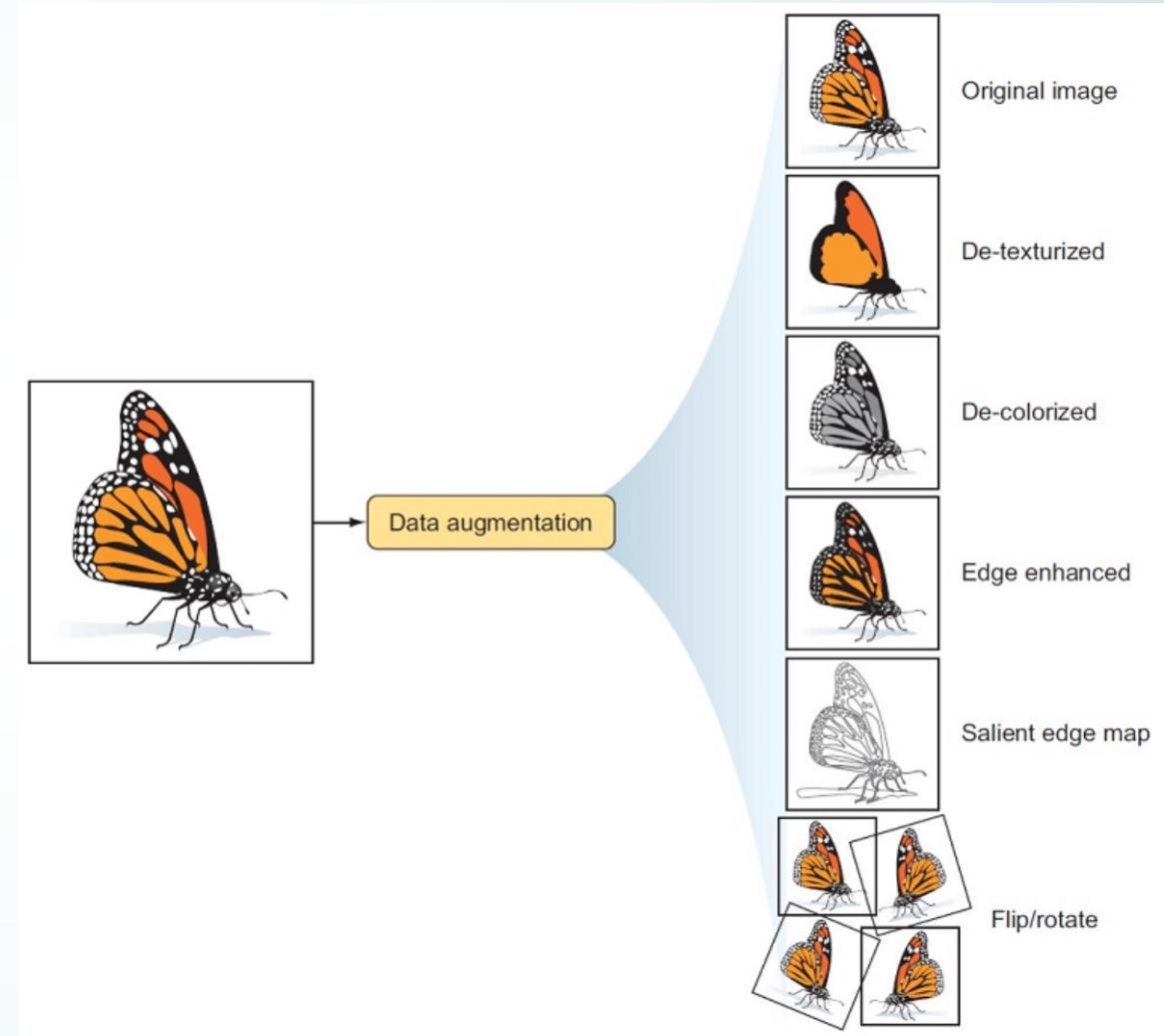
# Pipeline de vision par ordinateur

1. Entrée.
2. Pre-traitement.
- 3 Extraction de caractéristiques.
4. prédiction



# Augmentation de l'image

- ▶ **L'augmentation des données** utilise les échantillons de données disponibles pour produire les nouveaux, en appliquant des opérations d'image telles que la rotation, la mise à l'échelle, la traduction, etc. Cela rend notre modèle robuste aux changements d'intrants et conduit à une meilleure généralisation.



# Image transformations

## Deux grands types:

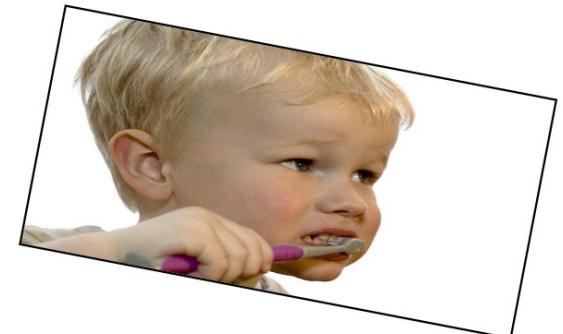
1. Filtres (Filtering)
2. Déformations (Warping)

Filtering



changes pixel *values*

Warping

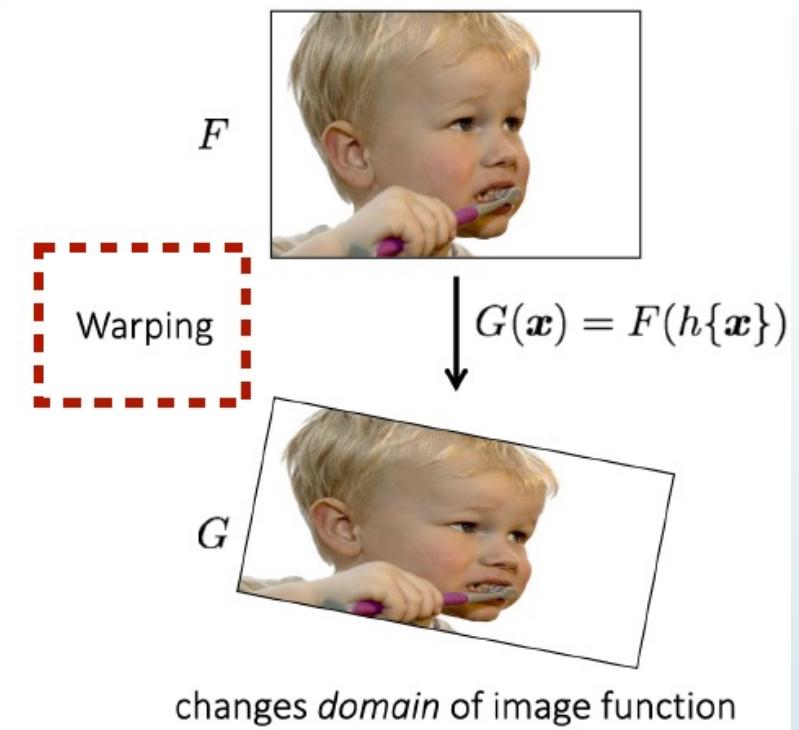
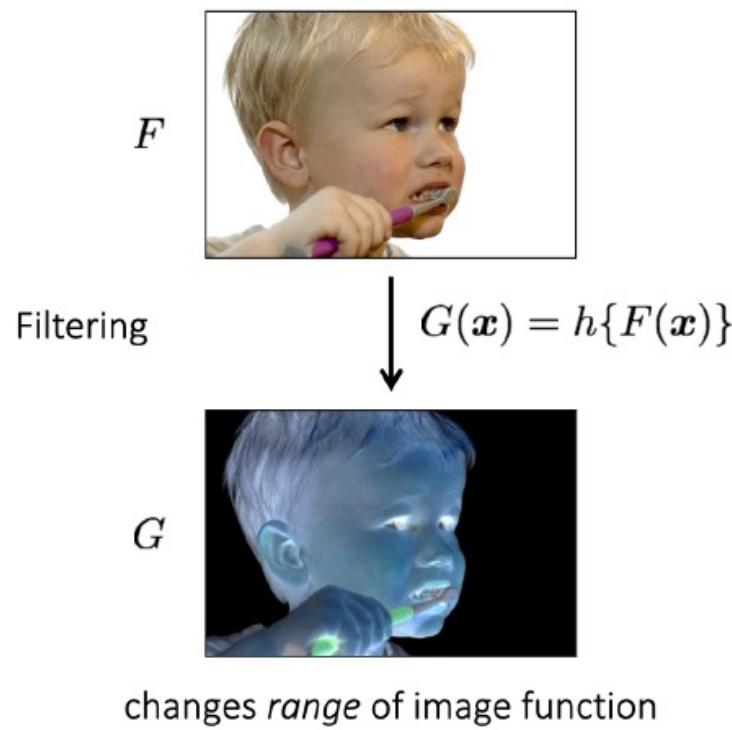


changes pixel *locations*

# Image transformations

## Deux grand types:

1. Filtres (Filtering)
2. Déformations (Warping)



# Transformation : Déformation

## Déformation:

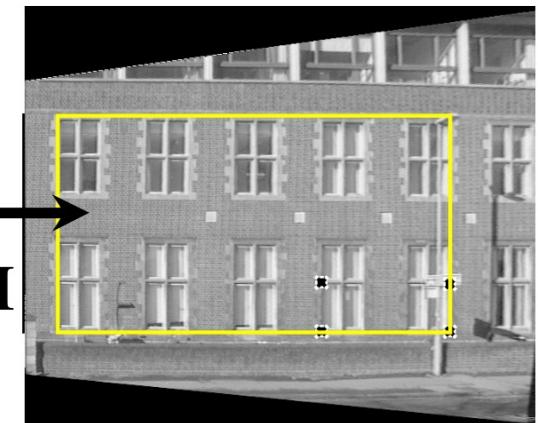
Ici: transformation projective  
(matrice de dimension 3 modifiant les  
coordonnées homogènes)  
Voir plus tard.

Transformation in this case is a projective transformation  
(general  $3 \times 3$  matrix, operating on homogeneous coords)



from Hartley & Zisserman

**Source Image**



**Destination image**

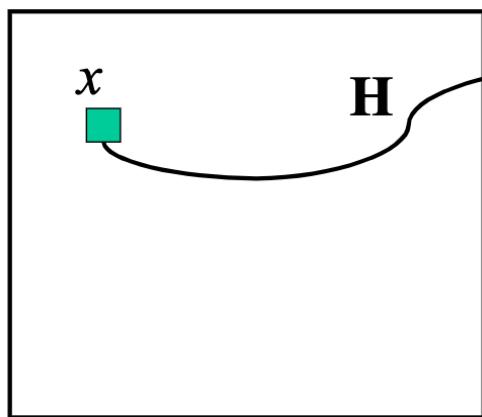
We will have a lot more to say about this in a future lecture.

# Déformation vers l'avant

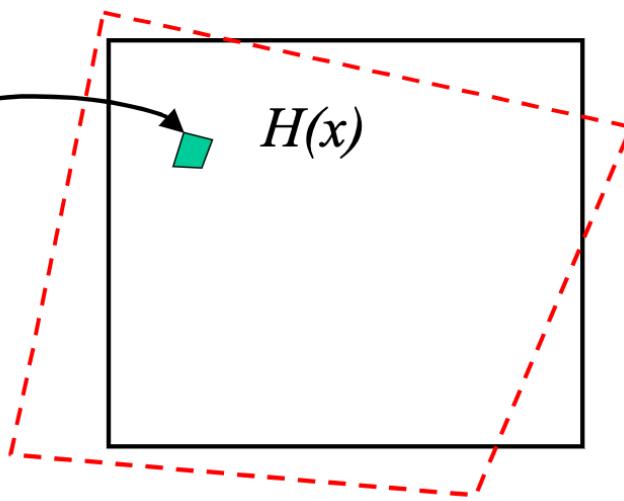
## Déformation vers l'avant:

- Pour chaque pixel  $(x, y)$  dans l'image source:
  - Déterminer sa position  $H(x, y)$
  - Colorier  $H(x, y)$  avec la couleur  $f(x, y)$   
 $f_{dest}(H(x, y)) = f_{source}(x, y)$
- Problème?

**Source image**



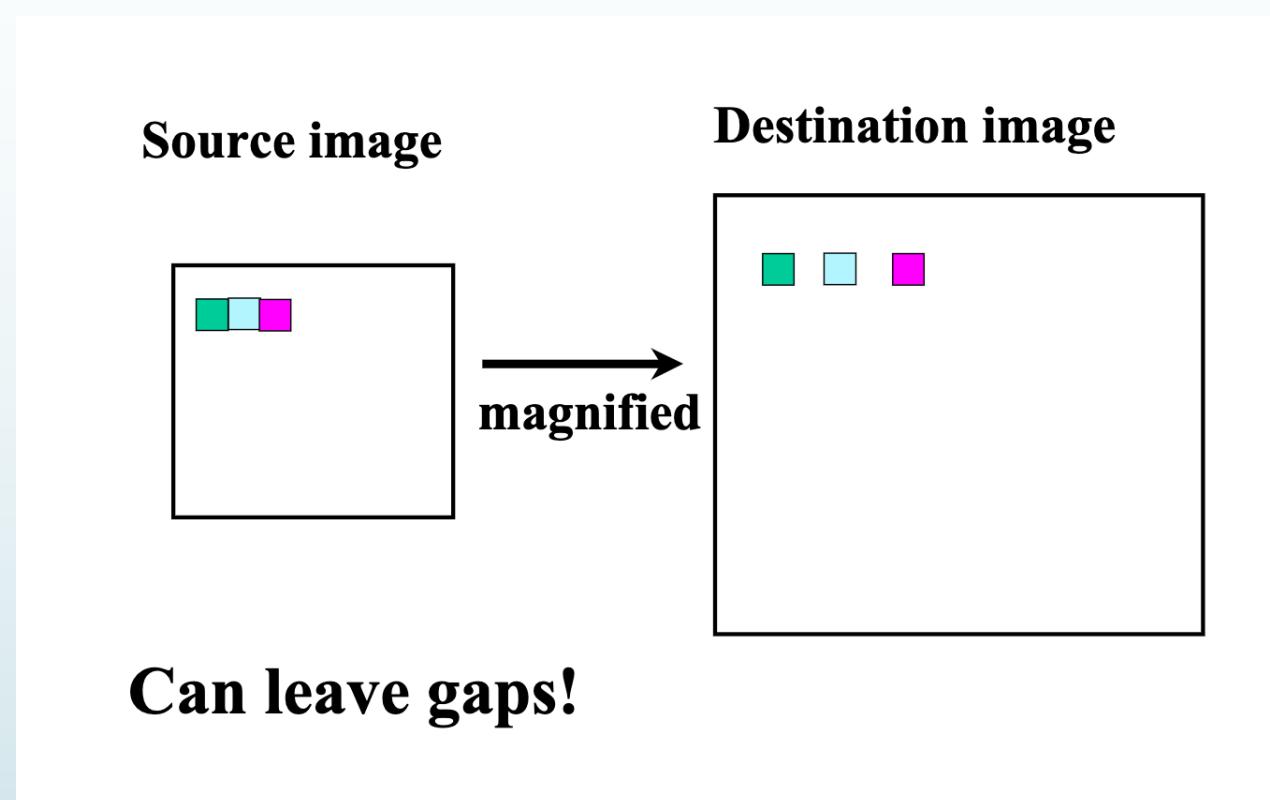
**Destination image**



- For each pixel  $x$  in the source image
- Determine where it goes as  $H(x)$
- Color the destination pixel

**Problems?**

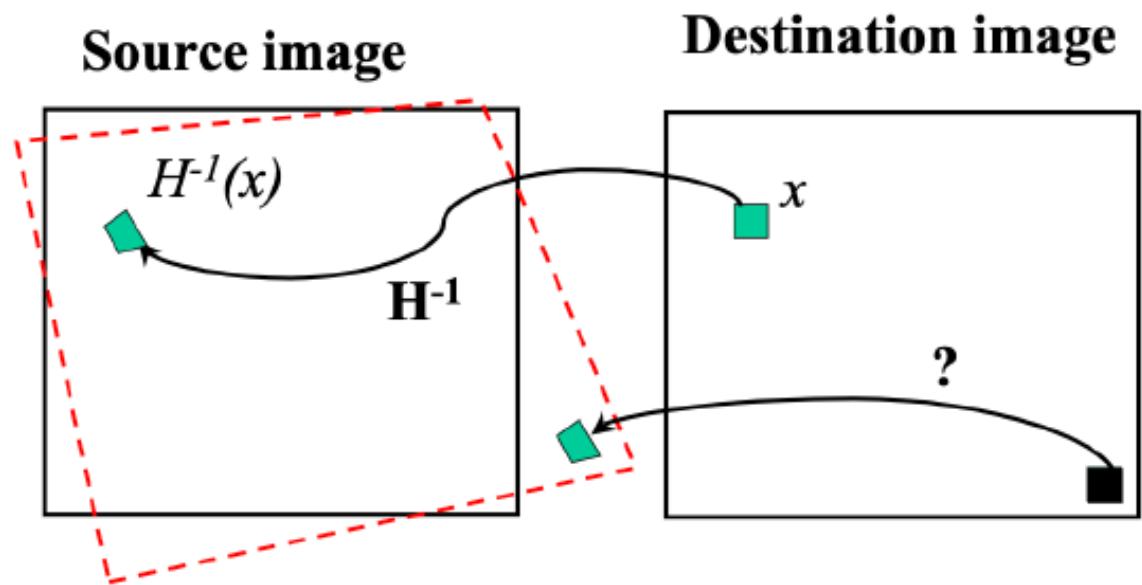
# Déformation vers l'avant (redimensionnement)



# Déformation vers l'arrière

## Déformation vers l'arrière:

- Pour chaque pixel  $(x, y)$  dans l'image destination:
  - Déterminer la position  $H^{-1}(x, y)$  dans l'image source.
  - Colorier  $(x, y)$  avec la couleur  $f(H^{-1}(x, y))$   
 $f_{dest}(x, y) = f_{source}(H^{-1}(x, y))$
- Utile quand la destination est "petite"

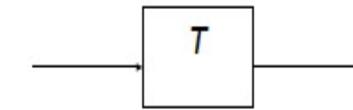


- For each pixel  $x$  in the destination image
- Determine where it comes from as  $H^{-1}(x)$
- Get color from that location

# Où vont les pixels ?



$p = (x, y)$



$p' = (x', y')$

- ▶ La transformation  $T$  change les coordonnées :  $p' = T(p)$ 
  - ▶ Qu'est-ce que cela signifie que  $T$  est global?
  - ▶ Est le même pour tout point  $p$
  - ▶ peut être décrit par quelques chiffres (paramètres)
  - ▶ Considérons les transformations linéaires (peuvent être représentées par une matrice 2D) :

$$p' = \mathbf{T}p \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Application linéaire

- Mise à l'échelle uniforme par s:

$$S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$



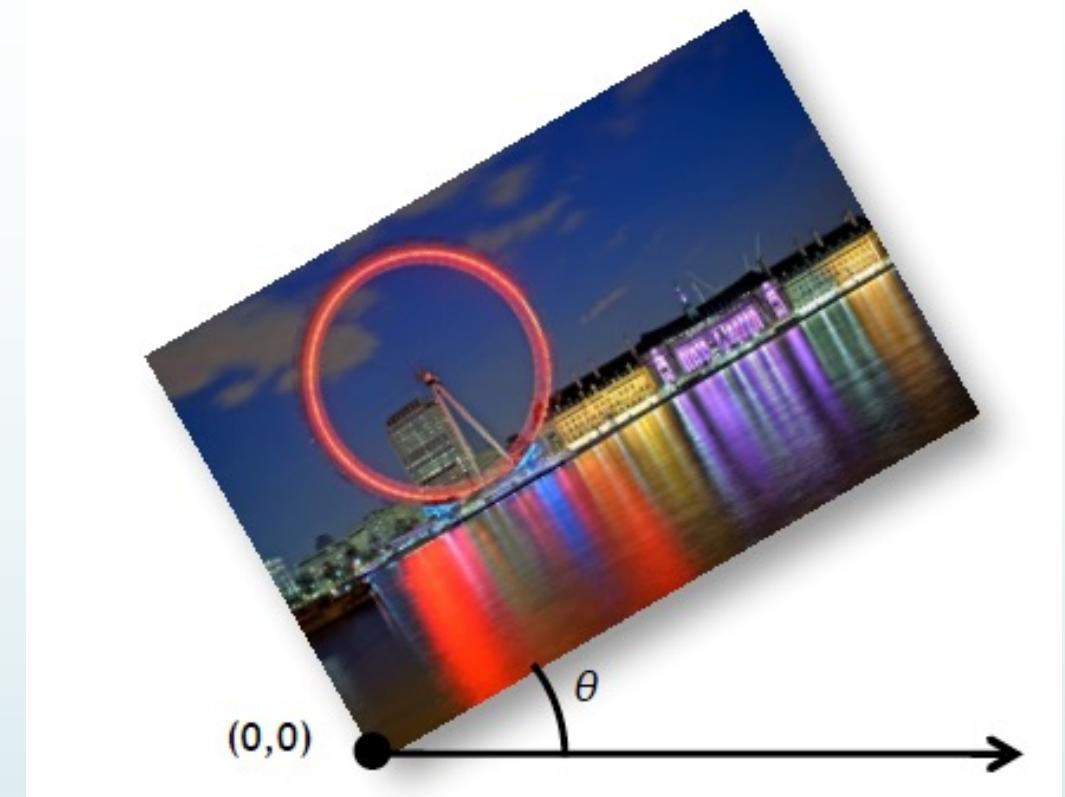
Quel est l'inverse ?

# Transformation Linéaire

- Rotation par angle  $\theta$  (autour de l'origine)



$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$



$$\mathbf{R}^{-1} = \mathbf{R}^T$$

- Quel est l'inverse ? Pour la rotation :

# Transformation avec matrices 2x2

- Quels types de transformations peuvent être représentés avec une matrice 2x2 ?
- 2D mirror about Y axis?

$$x' = -x$$

$$y' = y$$

$$\mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line  $y = x$ ?

$$x' = y$$

$$y' = x$$

$$\mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

# Transformation affine

- ▶ Les transformations affines sont des combinaisons de ...
  - ▶ Transformations linéaires, et
  - ▶ Translations
- ▶ Propriétés des transformations affines :
  - ▶ L'origine n'est pas nécessairement fixe
  - ▶ Les droites restent des droites
  - ▶ Les lignes parallèles restent parallèles
  - ▶ Les ratios sont préservés
  - ▶ Une composition de transformation affines est affine.

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

## Transformation affine

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

any transformation with last row [ 0 0 1 ] we call an *affine* transformation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

# Transformation de base

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D *in-plane* rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

# Transformation projective

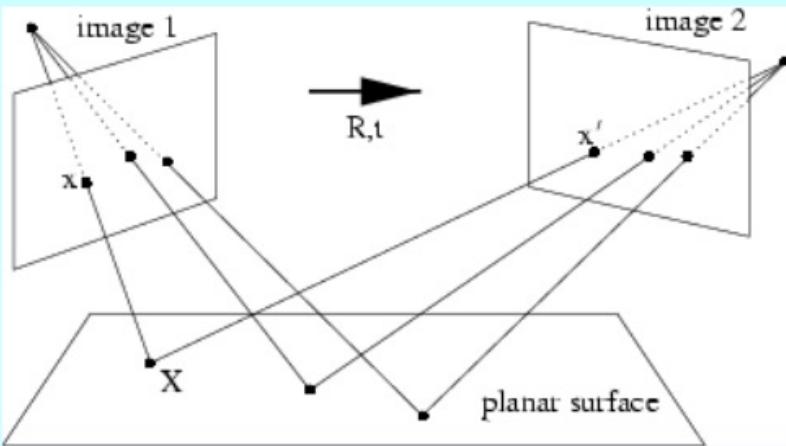
$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

affine transformation



what happens when we mess with this row?

# Projective transformation



$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*  
(or *planar perspective map*)



# Transformation projective

- ▶ Transformations projectives ...
  - ▶ Transformations affines, et
  - ▶ Déformations projectives

- ▶ Propriétés des transformations projectives:
  - ▶ L'origine n'est pas nécessairement fixe
  - ▶ Les droites restent des droites
  - ▶ Les lignes parallèles ne restent **pas forcément** parallèles
  - ▶ Les ratios **ne sont pas préservés**
  - ▶ Une composition de transformation projective est projective.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

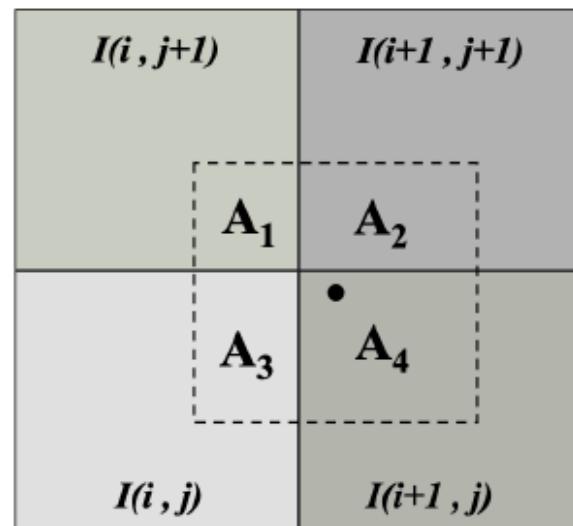
# Interpolation bilinéaire

Quelle est l'intensité au point  $(x, y)$ ?

Pas clair lorsque l'on est entre deux pixels.

**Interpolation:** calcule une **moyenne pondérée** des pixels voisins.

What do we mean by “get color from that location”? Consider grey values. What is intensity at  $(x, y)$ ?



**Bilinear Interpolation:**  
Weighted average

$$\begin{aligned} I(x,y) = & A_3 * I(i,j) \\ & + A_4 * I(i+1,j) \\ & + A_2 * I(i+1,j+1) \\ & + A_1 * I(i,j+1) \end{aligned}$$

Bilinear interpolation; the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood

# Interpolation linéaire (rappel)

Quelle est l'intensité au point  $(x, y)$ ?

Pas clair lorsque l'on est entre deux pixels.

**Interpolation:** calcule une **moyenne pondérée** des pixels voisins.

**Idée:** la fonction est linéaire entre  $(i,j)$  et  $(i+1,j)$

First, consider linear interpolation



Intuition: Given two pixel values, what should the value be at some intermediate point between them?



If close to  $(i,j)$ , should be intensity similar to  $I(i,j)$

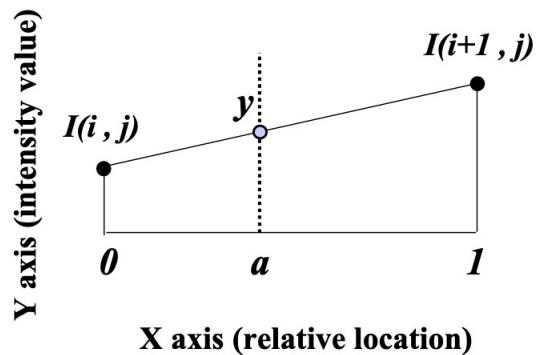
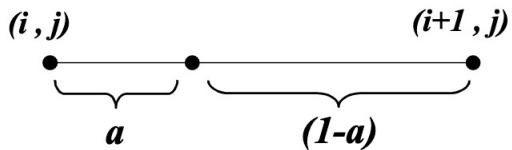


If equidistant from both, should be average of the two intensities



If close to  $(i+1,j)$ , should be intensity similar to  $I(i+1,j)$

# Interpolation linéaire (rappel)



Recall:

$$y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

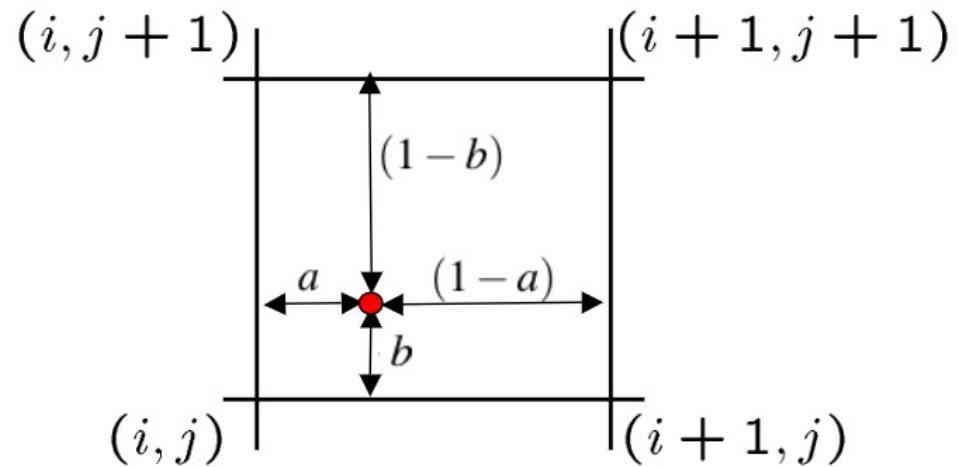
Instantiate

$$y - I(i, j) = \frac{(I(i+1, j) - I(i, j))}{(1 - 0)}(a - 0)$$

Solve

$$y = (1 - a) I(i, j) + a I(i + 1, j)$$

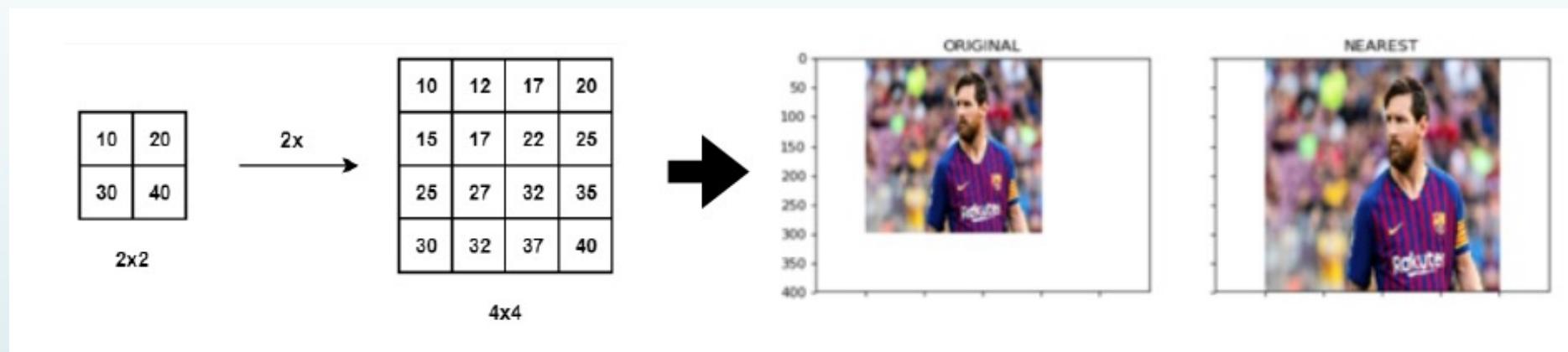
# Interpolation bilinéaire



$$\begin{aligned}\mathbf{I} = & (1-a)(1-b) I(i, j) \\ & + a (1-b) I(i+1, j) \\ & + (1-a) b I(i, j+1) \\ & + a b I(i+1, j+1)\end{aligned}$$

# Redimensionnement de l'image

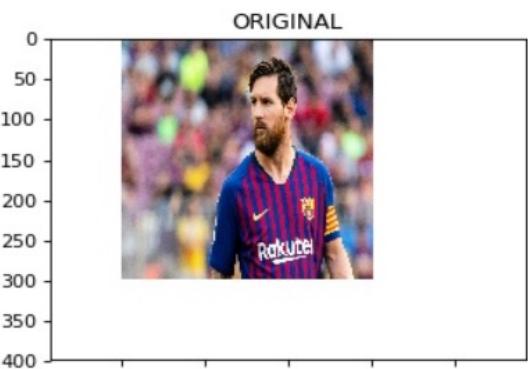
- ▶ Les modèles d'apprentissage automatique fonctionnent avec une **entrée de taille fixe**. La même idée s'applique également aux modèles de vision par ordinateur. Les images que nous utilisons pour entraîner notre modèle doivent être de la même taille.
- ▶ Les images peuvent être facilement mises à l'échelle:



- ▶ Différentes méthodes d'interpolation et de sous-échantillonnage sont prises en charge par **OpenCV**. La fonction **resize** d'OpenCV utilise l'interpolation bilinéaire par défaut.

# Redimensionnement avec OpenCV

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5 #reading the image
6 image = cv2.imread('index.jpg')
7 #converting image to size (100,100,3)
8 smaller_image = cv2.resize(image,(100,100),interpolation='linear')
9 #plot the resized image
10 plt.imshow(smaller_image)
```



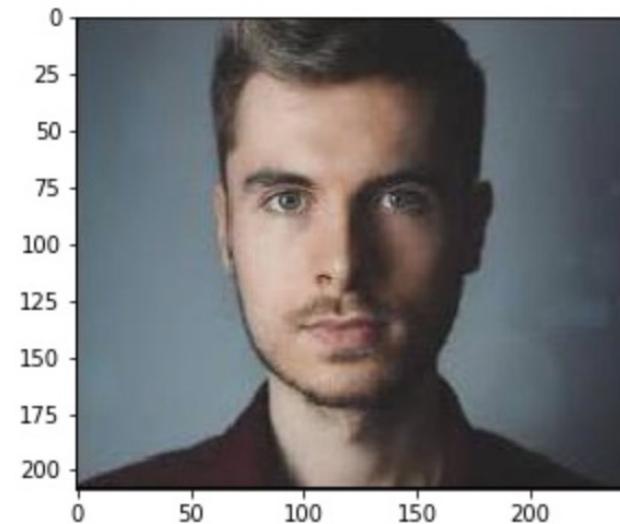
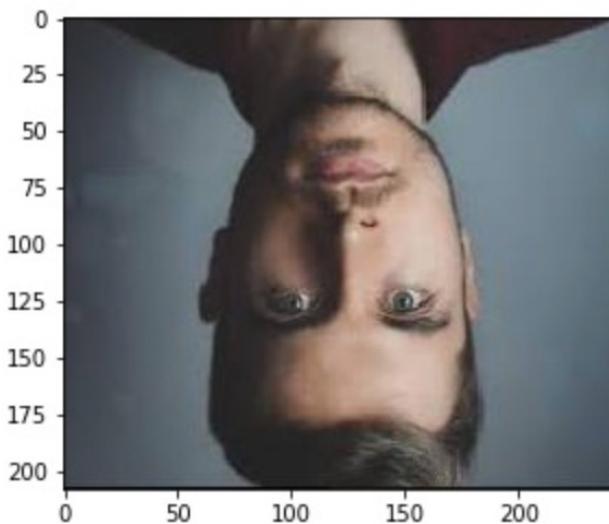
# Rotation

- ▶ Supposons que nous construisions un modèle de classification d'image pour identifier l'animal présent dans une image. Ainsi, les deux images ci-dessous doivent être classées comme « chien »:



# Rotation avec OpenCV

```
1 #importing the required libraries
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 image = cv2.imread('index.png')
7 rows,cols = image.shape[:2]
8 #(col/2,rows/2) is the center of rotation for the image
9 # M is the coordinates of the center
10 M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
11 dst = cv2.warpAffine(image,M,(cols,rows))
12 plt.imshow(dst)
```



# Déplacement

```
1 #importing the required libraries
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 #reading the image
7 image = cv2.imread('index.png')
8 #shifting the image 100 pixels in both dimensions
9 M = np.float32([[1,0,-100],[0,1,-100]])
10 dst = cv2.warpAffine(image,M,(cols,rows))
11 plt.imshow(dst)
```



# Resources

- ▶ [CV course CMU](#)
- ▶ [CV course Stanford](#)
- ▶ [CV course Penn State](#)