UNIVERSITÉ LAVAL

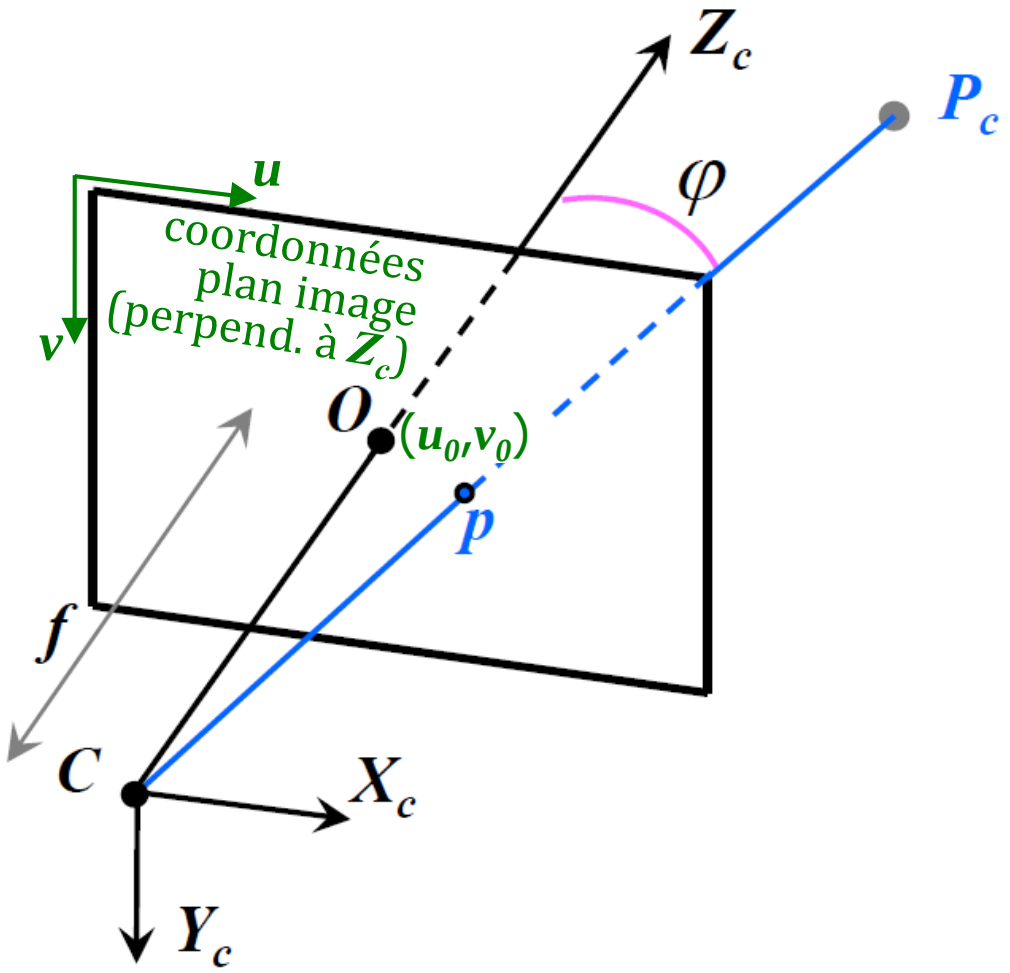
GLO-4001/7021
INTRODUCTION À LA ROBOTIQUE MOBILE

Capteurs Visuels II

(Automne 2019)

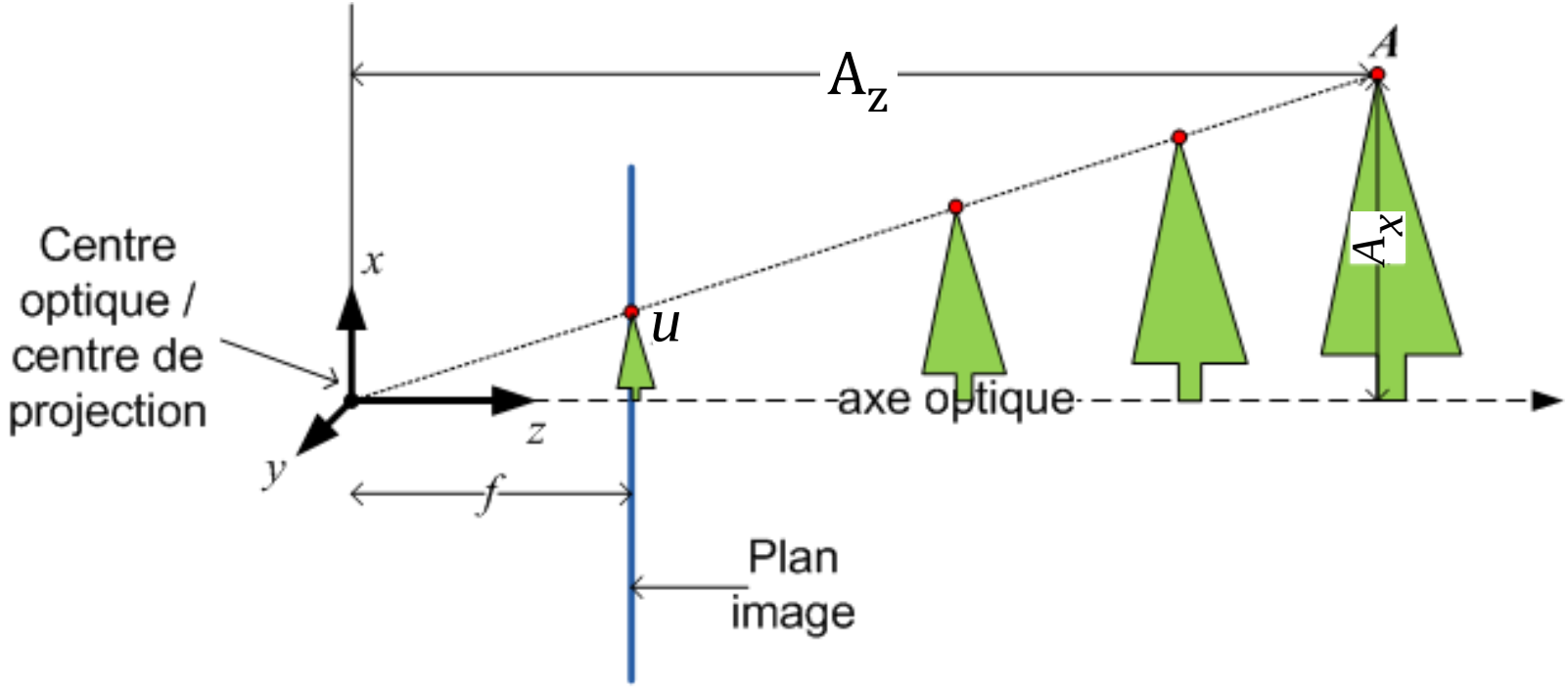
Philippe Giguère

Rappel : caméra sténopé



Rappel : perte de 3D

- Problème mal posé (plusieurs solutions pour l'inverse)



Perte de l'échelle des dimensions physiques (m)

Rappel : équation caméra perspective

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \\ 1 \end{bmatrix}$$

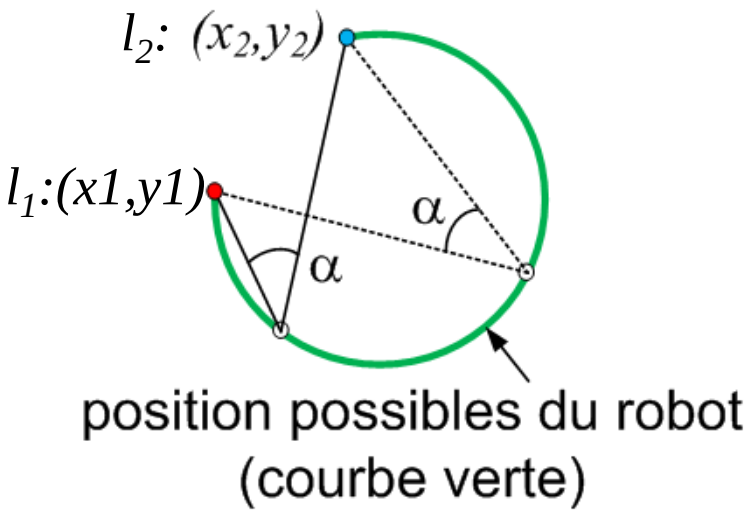
coordonnées homogènes

Retrouve la position dans le plan image : $(x_1/x_3, x_2/x_3)$

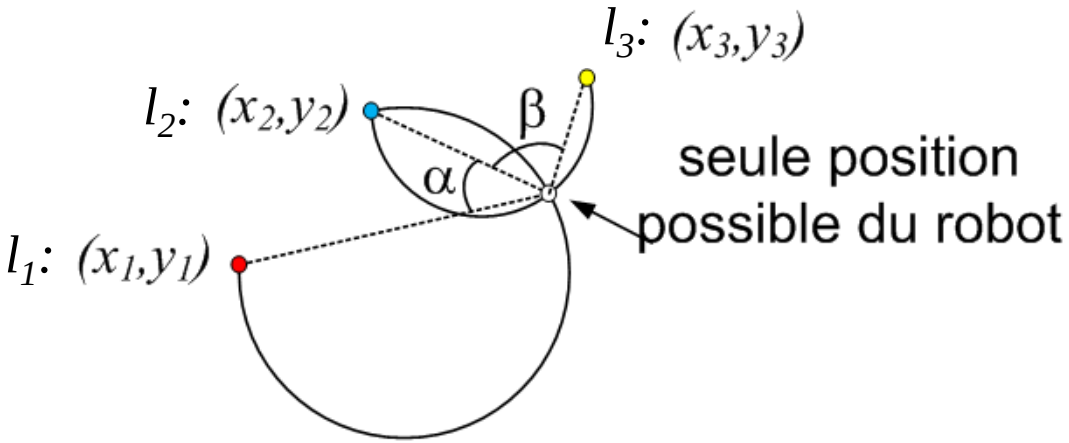
Localisation 2D par triangulation (angles)

- Angles α et β entre pt. de repères dans l'image de la caméra
- Connait position des points de repères l_1, l_2 et l_3

Deux repères, un angle α.



Trois repères, deux angles α, β.



manuel Comp. Princ.
p. 243-246

Imagerie stéréo : retrouver la 3D

Images stéréo

- 2 caméras placés à b=5 cm distance, axes optiques parallèles

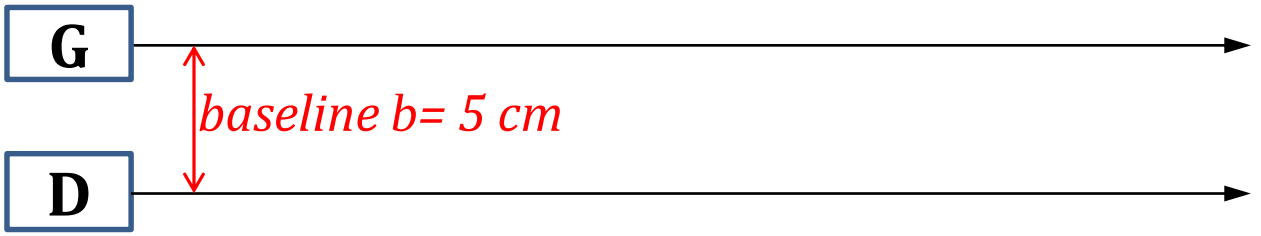
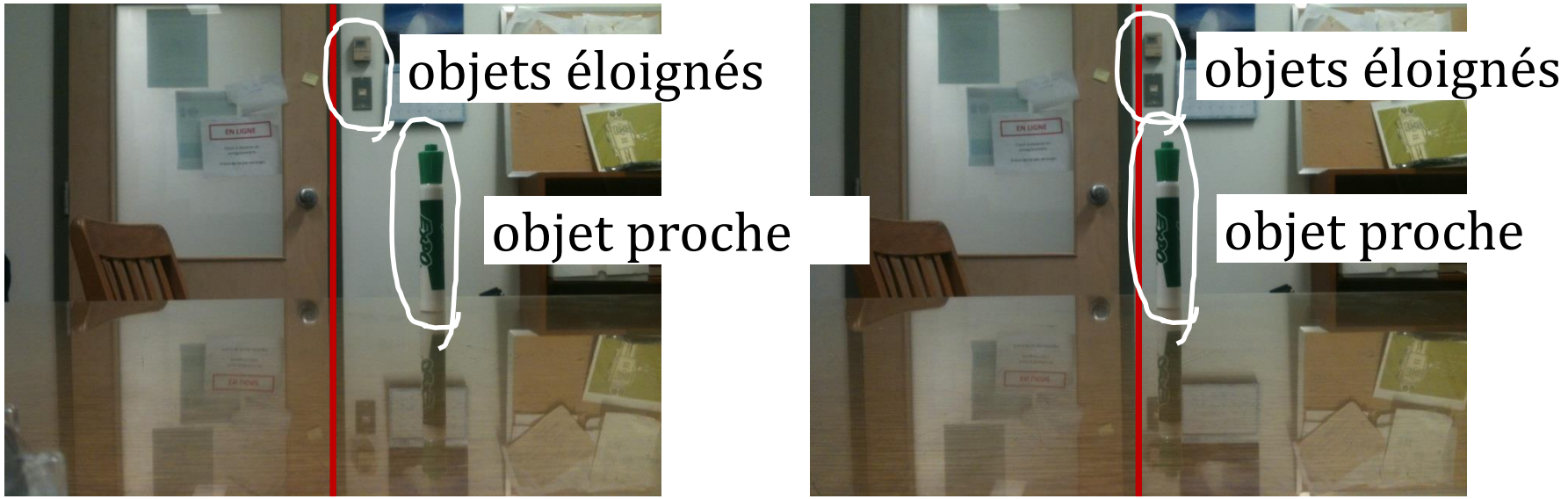


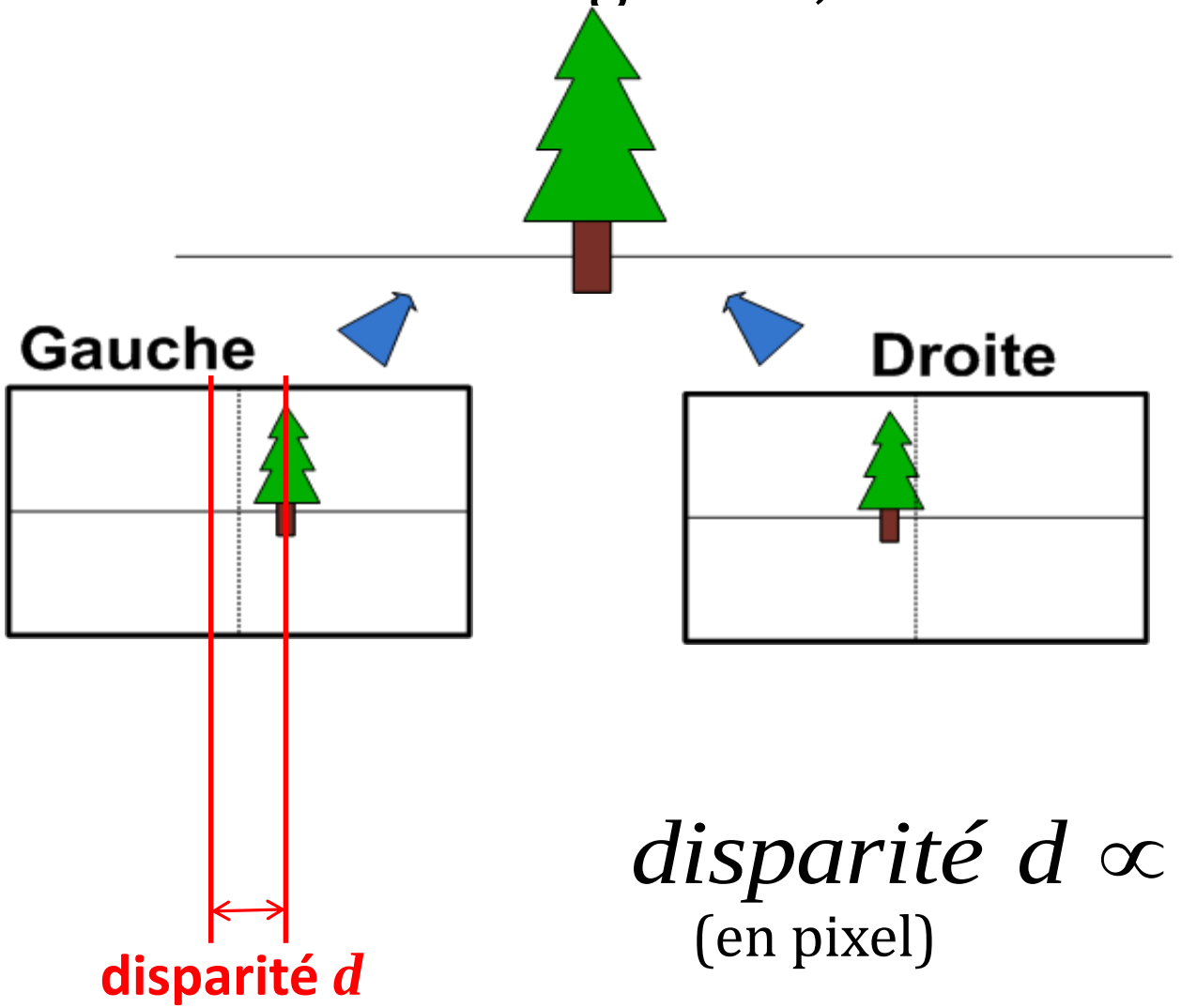
Image de caméra gauche

Image de caméra droite



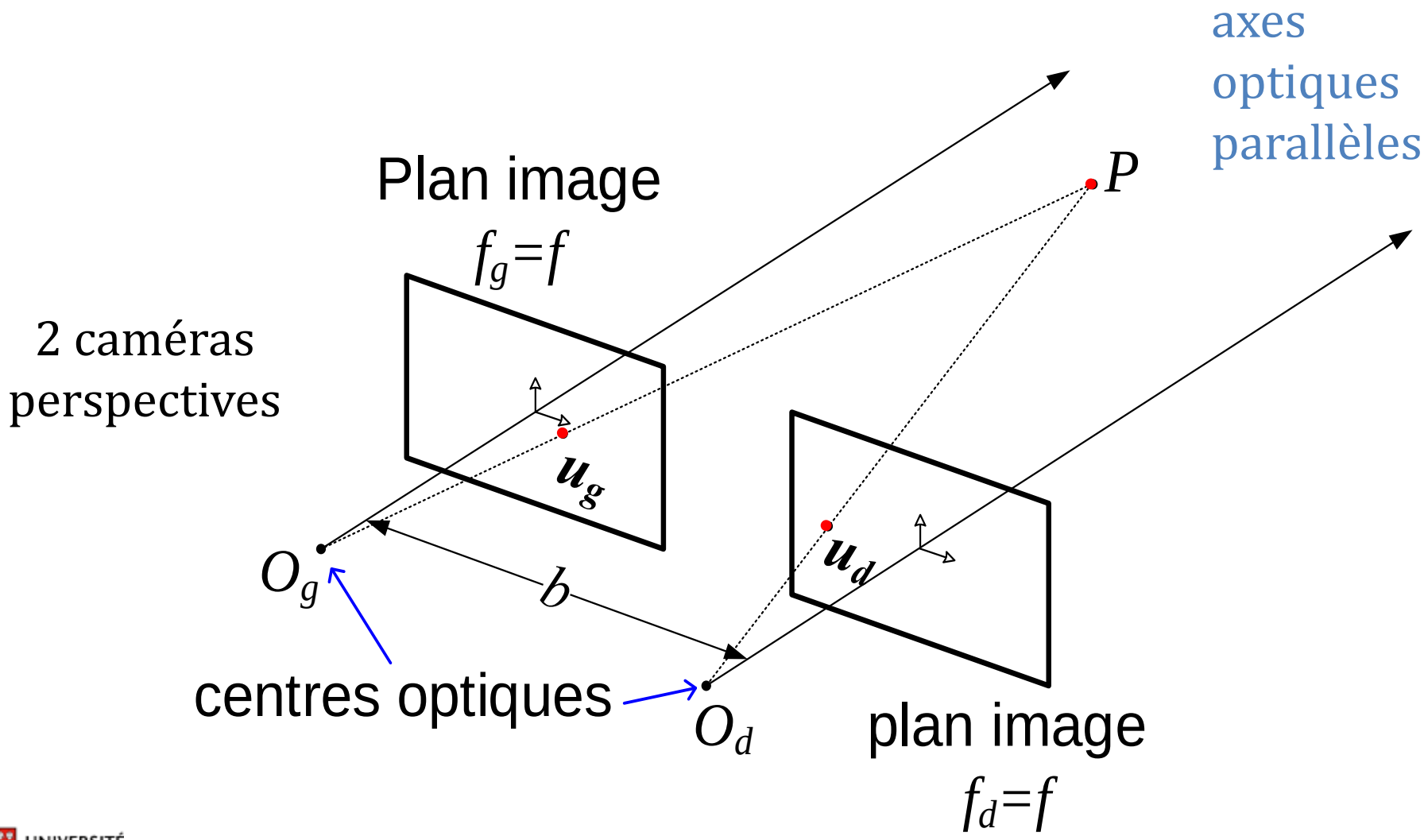
Imagerie stéréo

- À partir de deux images 2D, retrouver 3D

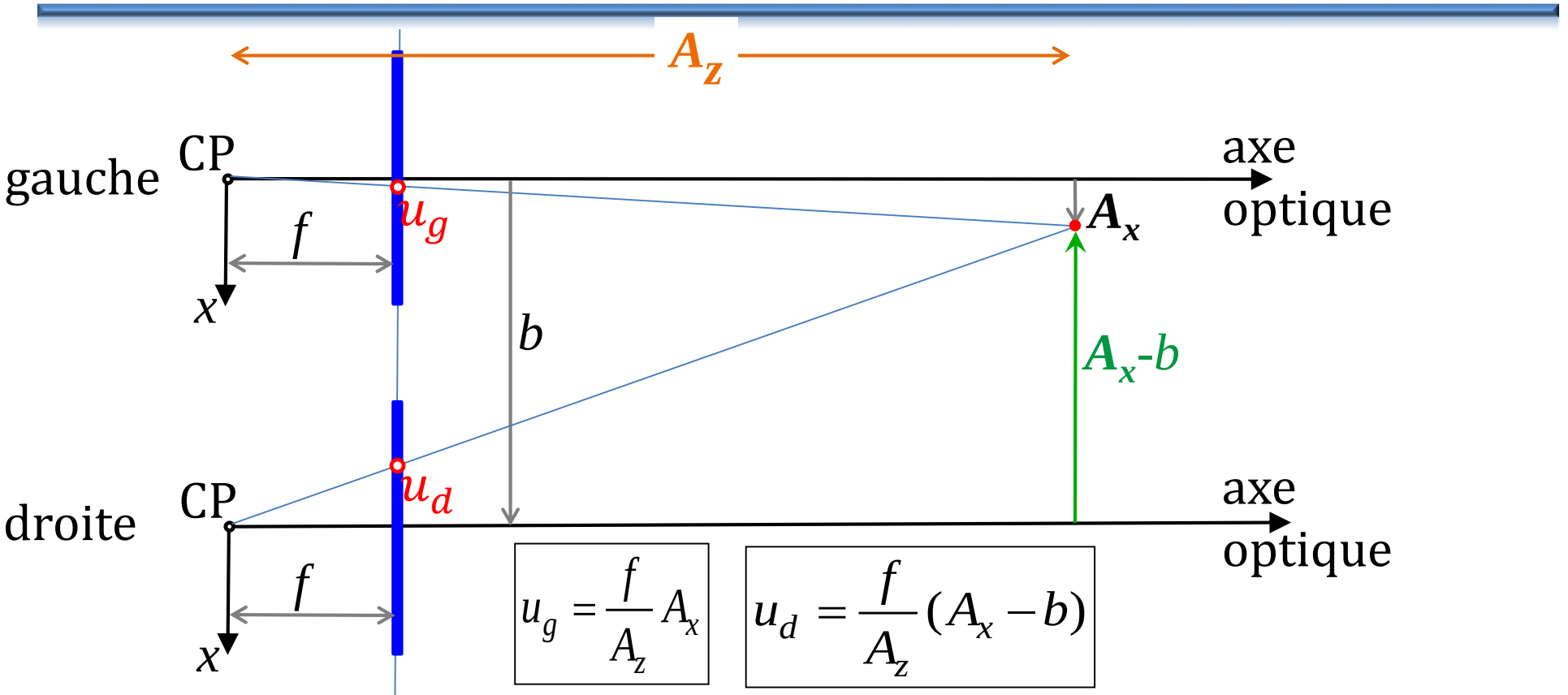


$$disparité\ d \propto {}^{1}\!/_{A_z}$$
(en pixel)

Caméra stéréo



Disparité



$$u_g = \frac{f}{A_z} A_x \qquad u_d = \frac{f}{A_z}(A_x - b)$$

note: même focale f pour g et d

plans images

Disparité $d = u_g - u_d = \frac{f}{A_z} A_x - \frac{f}{A_z}(A_x - b) - = \frac{f}{A_z} b$
(définition)

On retrouve donc la profondeur du point : $A_z = \frac{f\ b}{d}$

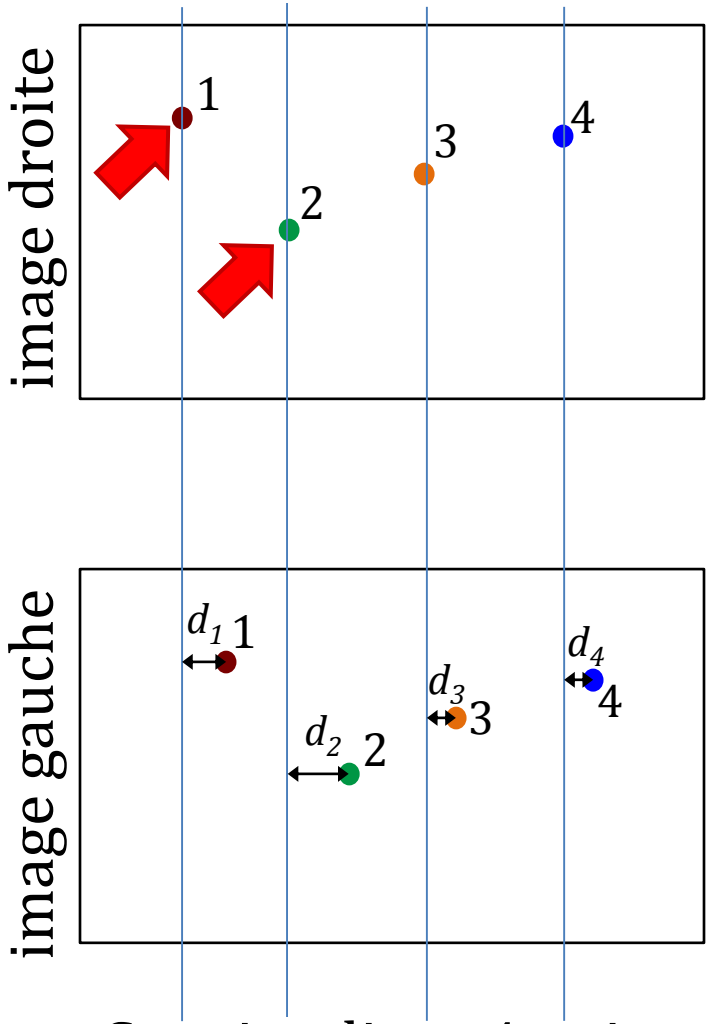
Sensibilité stéréo : *baseline b*

Télémètre Allemand
Entfernungsmesser-1m-R-36
Tir anti-aérien
Baseline $\boldsymbol{b}$= 1 *m*

$$A_z = \frac{f\ b}{d}$$

Disparité à chaque point visible (pixel?)

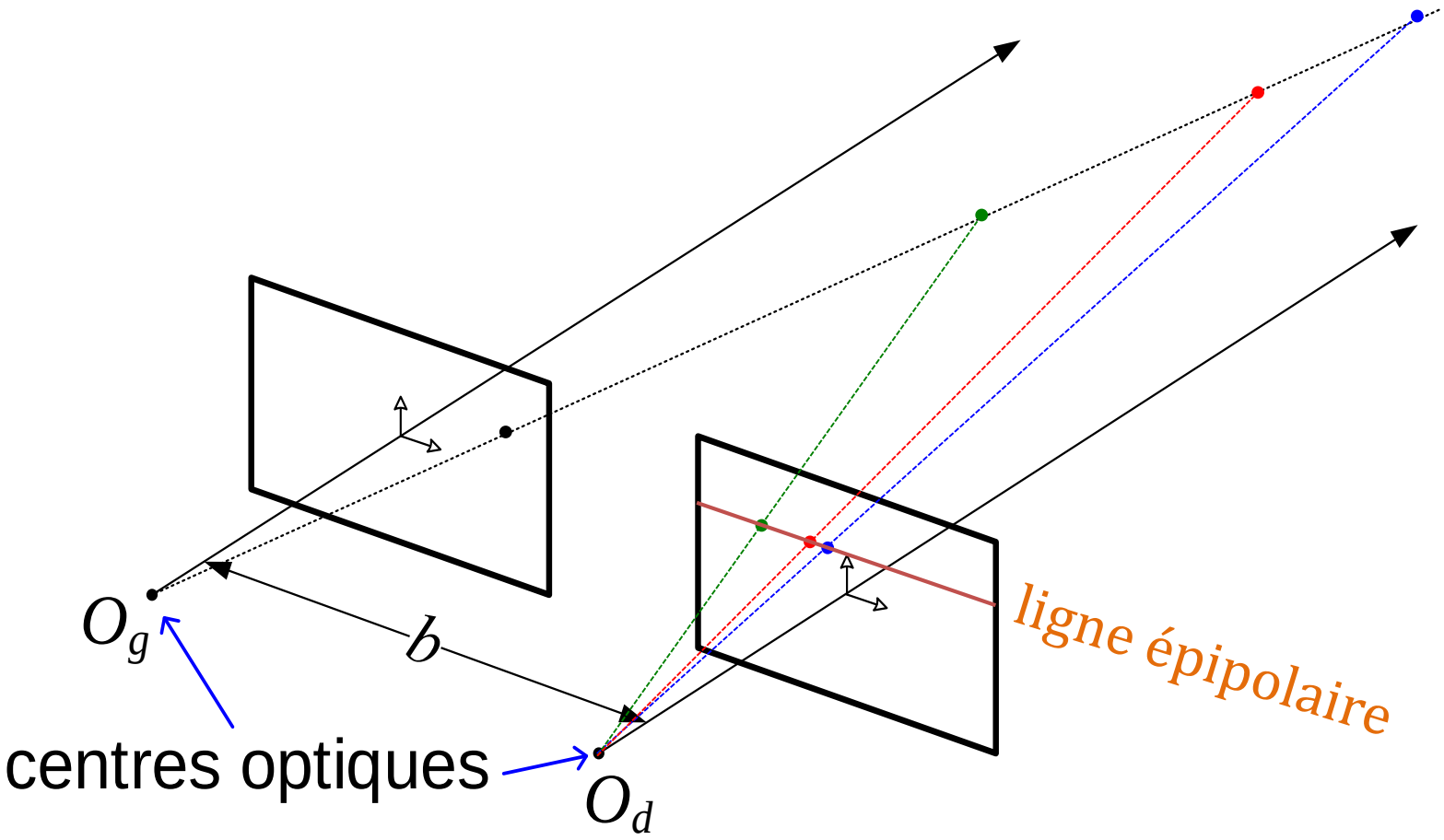
- Pour chaque point visible i, on doit :
  - faire la correspondance entre les deux images *(basé sur l'apparence)* *data association*
  - estimer la disparité d_i
- On pourra ainsi retrouver la profondeur A_{zi} pour chaque point i
- Disparité d_i pour chaque pixel : *depth map*



Cas simpliste, 4 points visibles devant la caméra

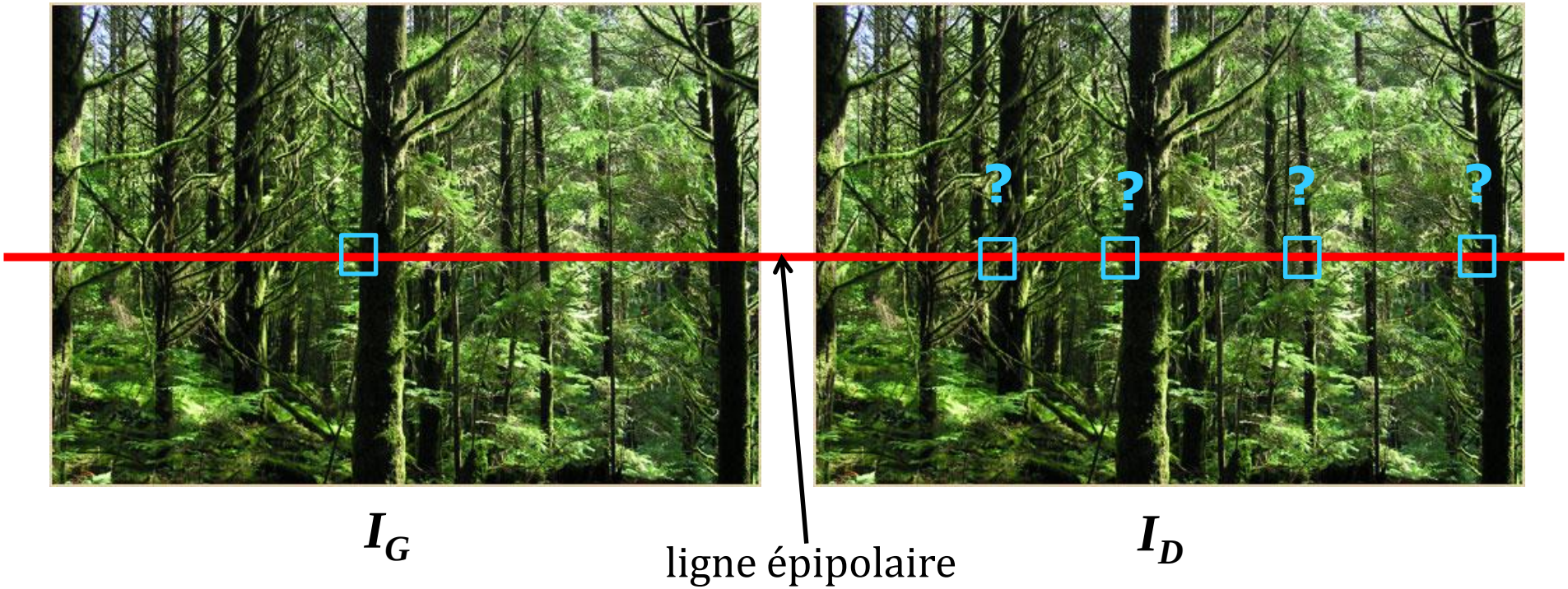
UNIVERSITÉ LAVAL

Ligne épipolaire



Problème de correspondance… *data association*

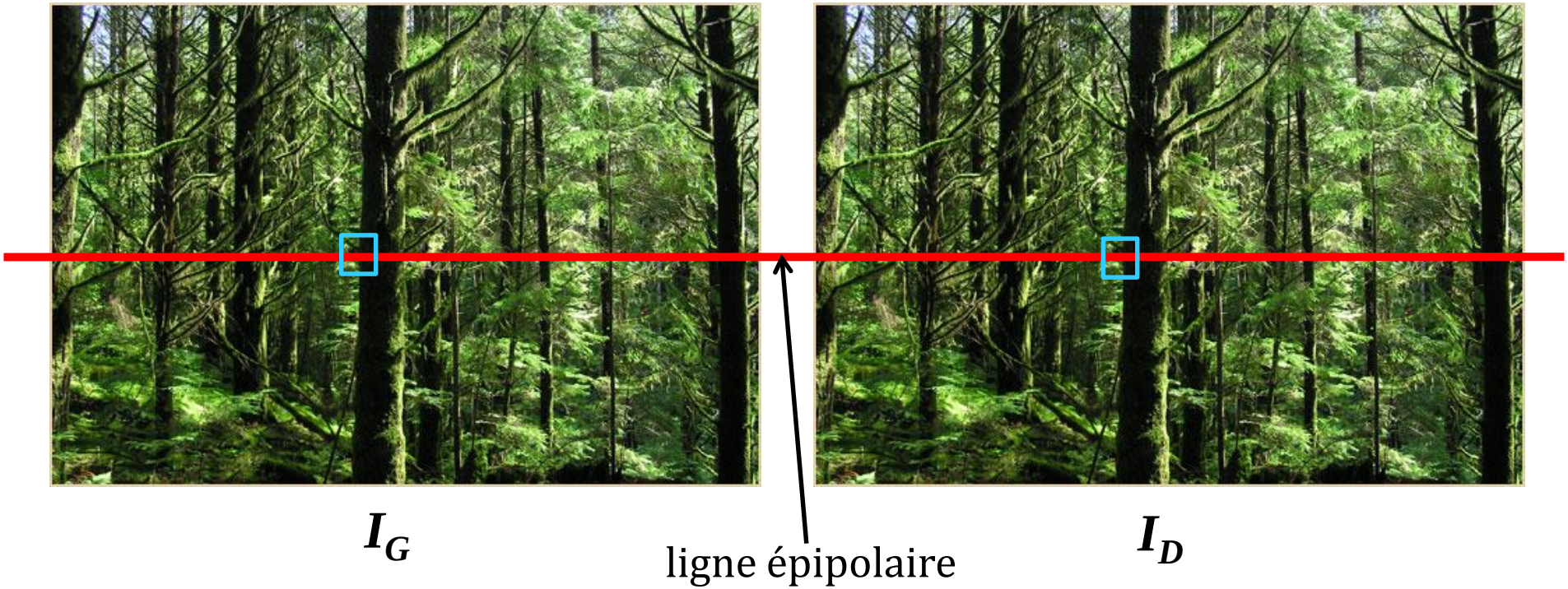
- Quel pixel dans l'image I_D correspond au pixel du même point physique dans image I_G sur la **ligne épipolaire***?



*On assume que les caméras sont décalées strictement horizontalement, d'où la ligne épipolaire horizontale

Problème de correspondance… *data association*

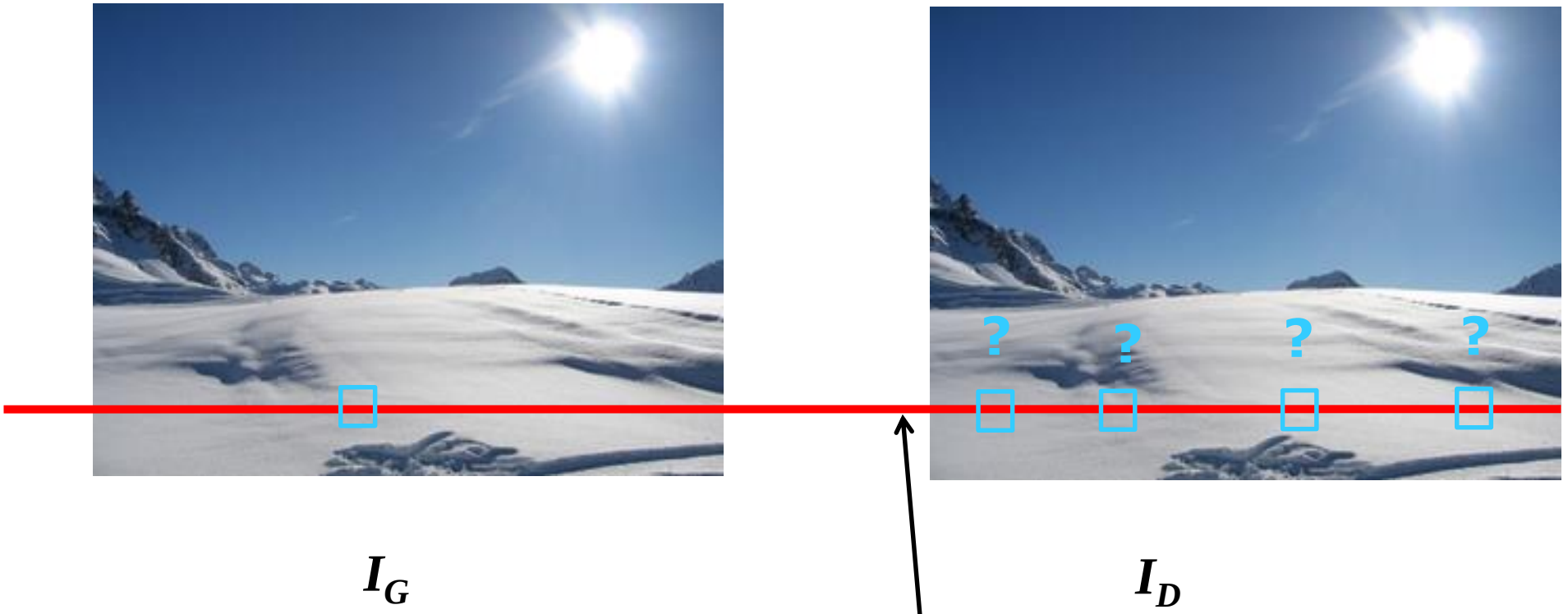
- Quel pixel dans l'image I_D correspond au pixel du même point physique dans image I_G sur la **ligne épipolaire***?



*On assume que les caméras sont décalées strictement horizontalement, d'où la ligne épipolaire horizontale

Problème de correspondance…

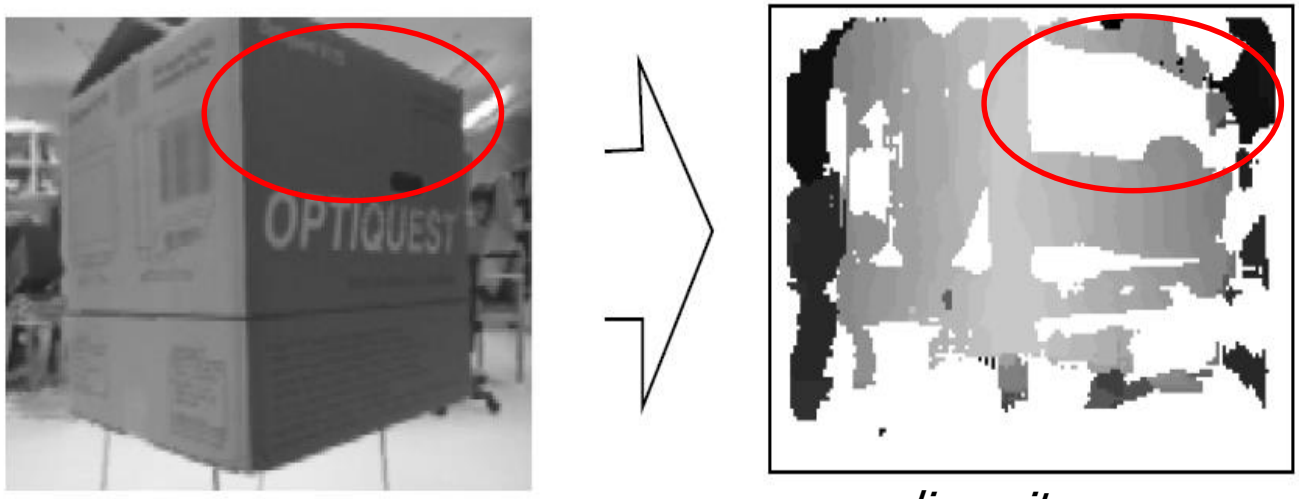
- Quel point dans image I_D correspond au point dans l'image I_G?



Exemples *disparity map*

- *Disparity map* : valeur de A_z pour chaque pixel

manque *features* visuels



disparity map

Using real-time stereo vision for mobile robot navigation

Don Murray Jim Little

Computer Science Dept.
University of British Columbia
Vancouver, BC, Canada V6T 1Z4

Caméras actives

Kinect 1 : stéréo active

- Caméra « 3D »
- Lumière structurée

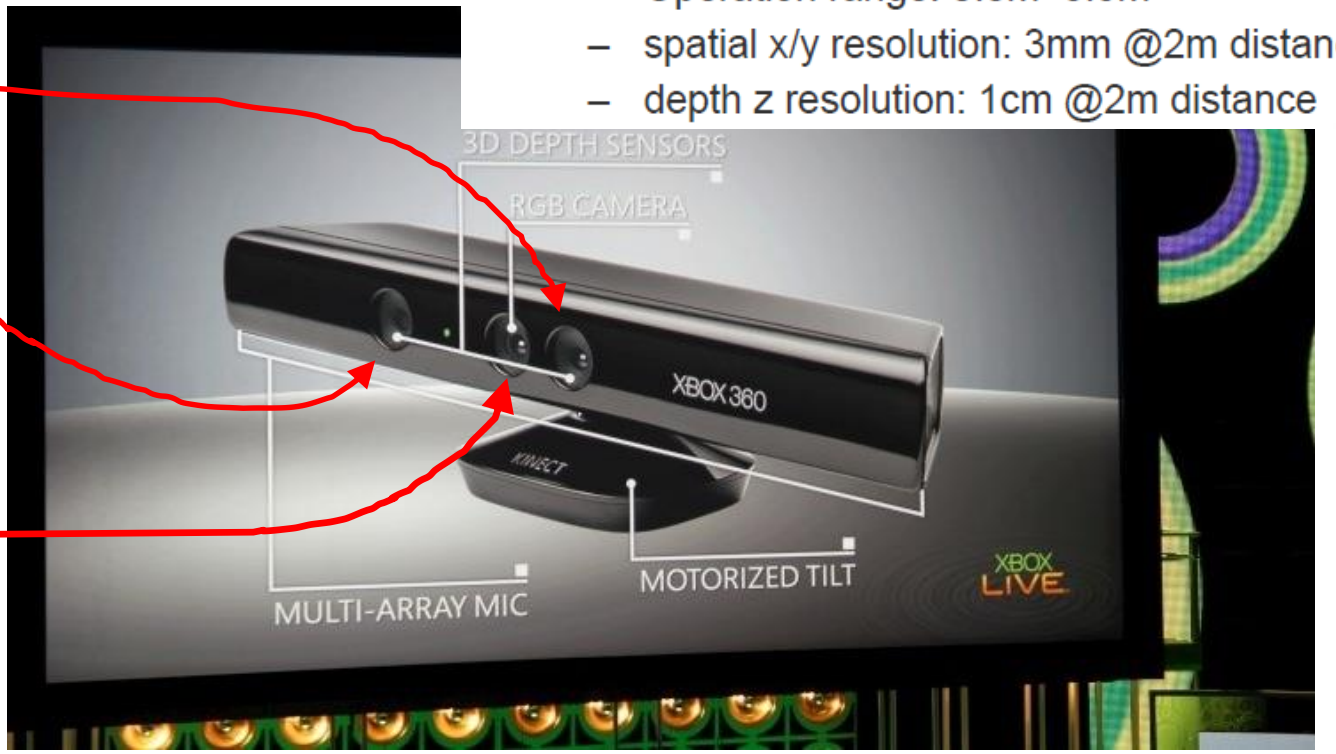
Microsoft Kinect

- Depth resolution: 640x480 px
- RGB resolution: 1600x1200 px
- 60 FPS
- Operation range: 0.8m~3.5m
- spatial x/y resolution: 3mm @2m distance
- depth z resolution: 1cm @2m distance

caméra infrarouge

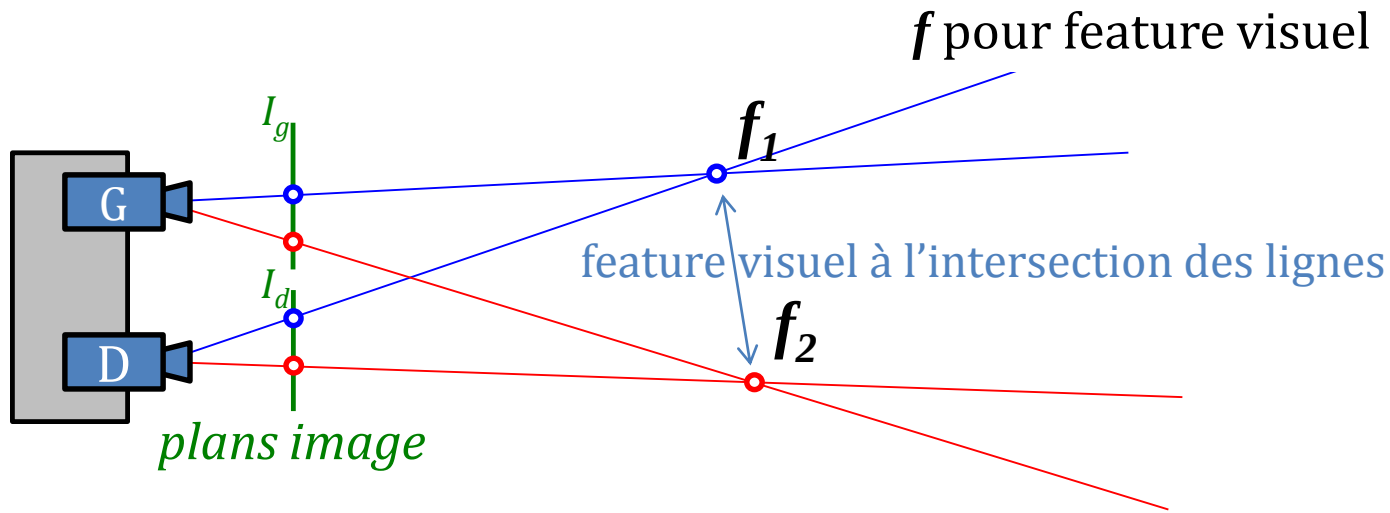
projecteur infrarouge

caméra RGB

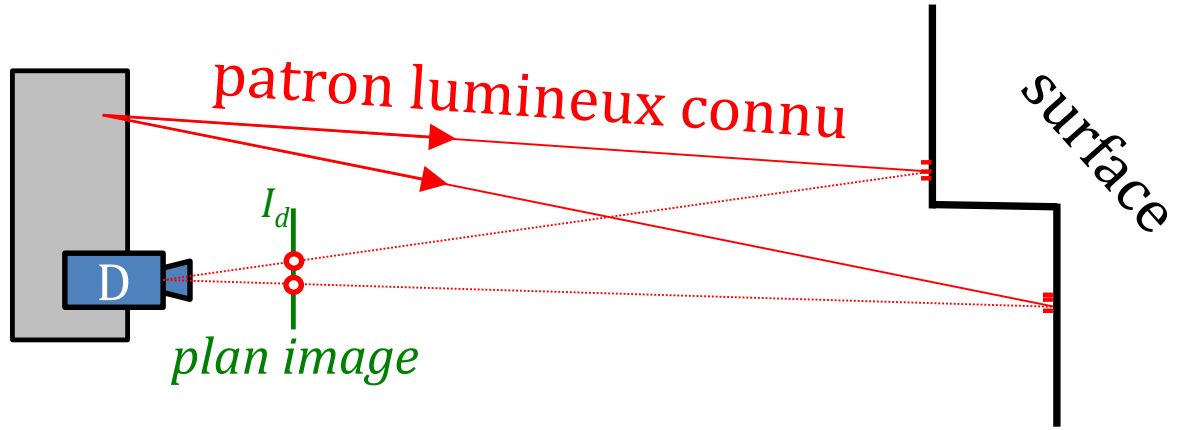


Kinect 1 : stéréo active

Stéréo normale



Stéréo active



Kinect 1 : stéréo active

- Projette patron infrarouge <u>localement distinct</u>
  (pensez code barre)

  Moins de problème de correspondance!

- Observe la « disparité » *d* avec la caméra infrarouge

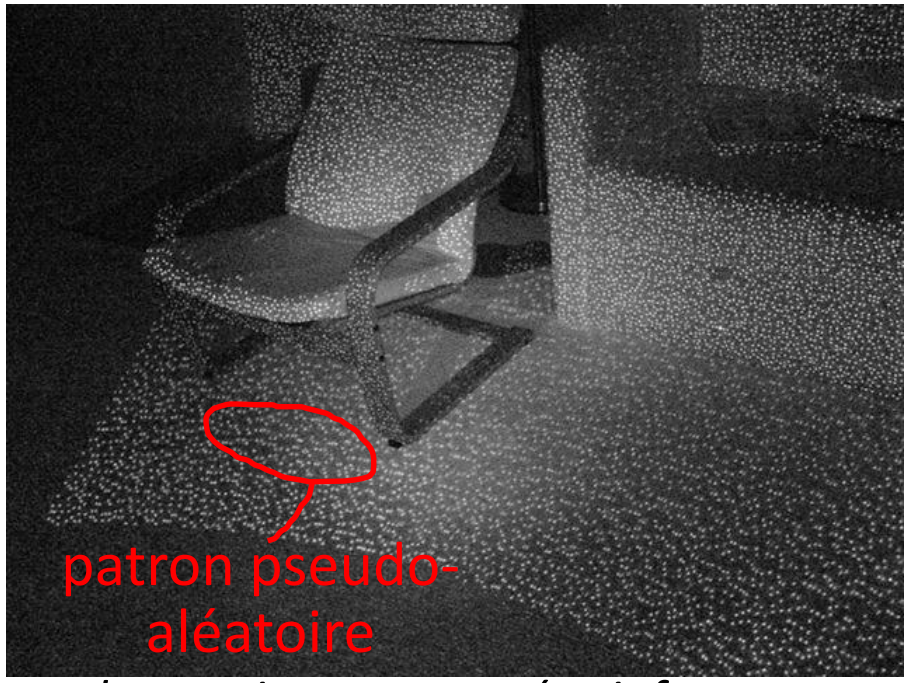
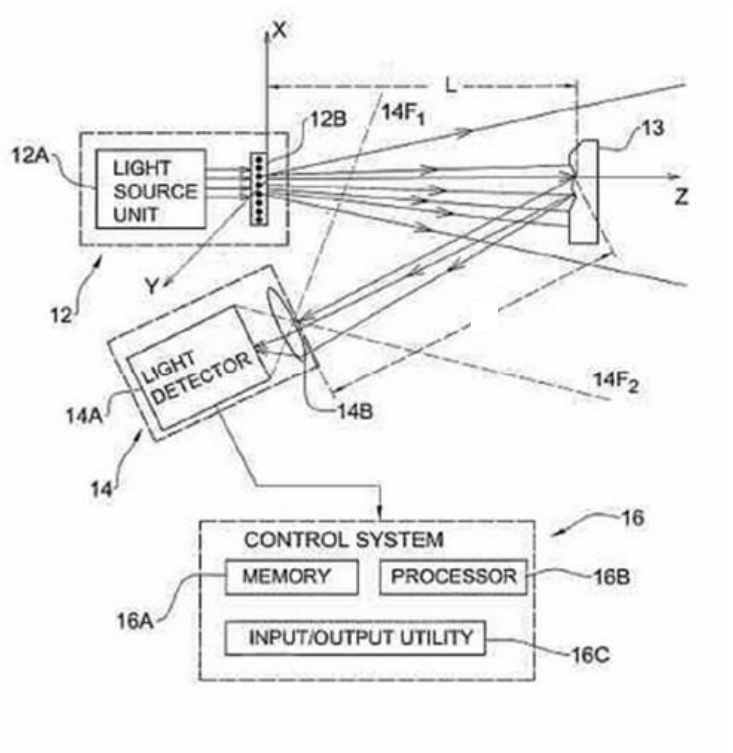


photo prise avec caméra infra-rouge



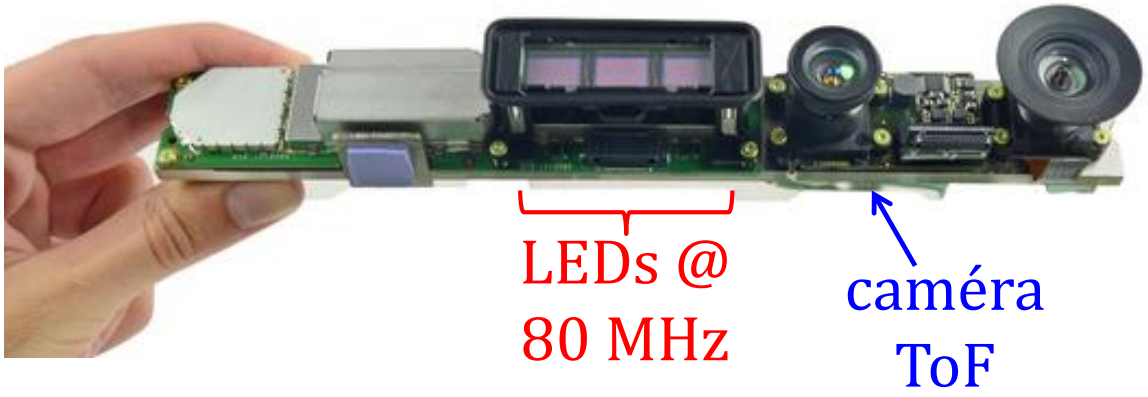
Kinect 1 : depth image

- Retourne une image de profondeur (*depth image*)
- Pour chaque pixel, on aura la distance en *Z* (ou rien)
- 640x480 pixels, 30 *Hz*
- Précision dépend de la distance

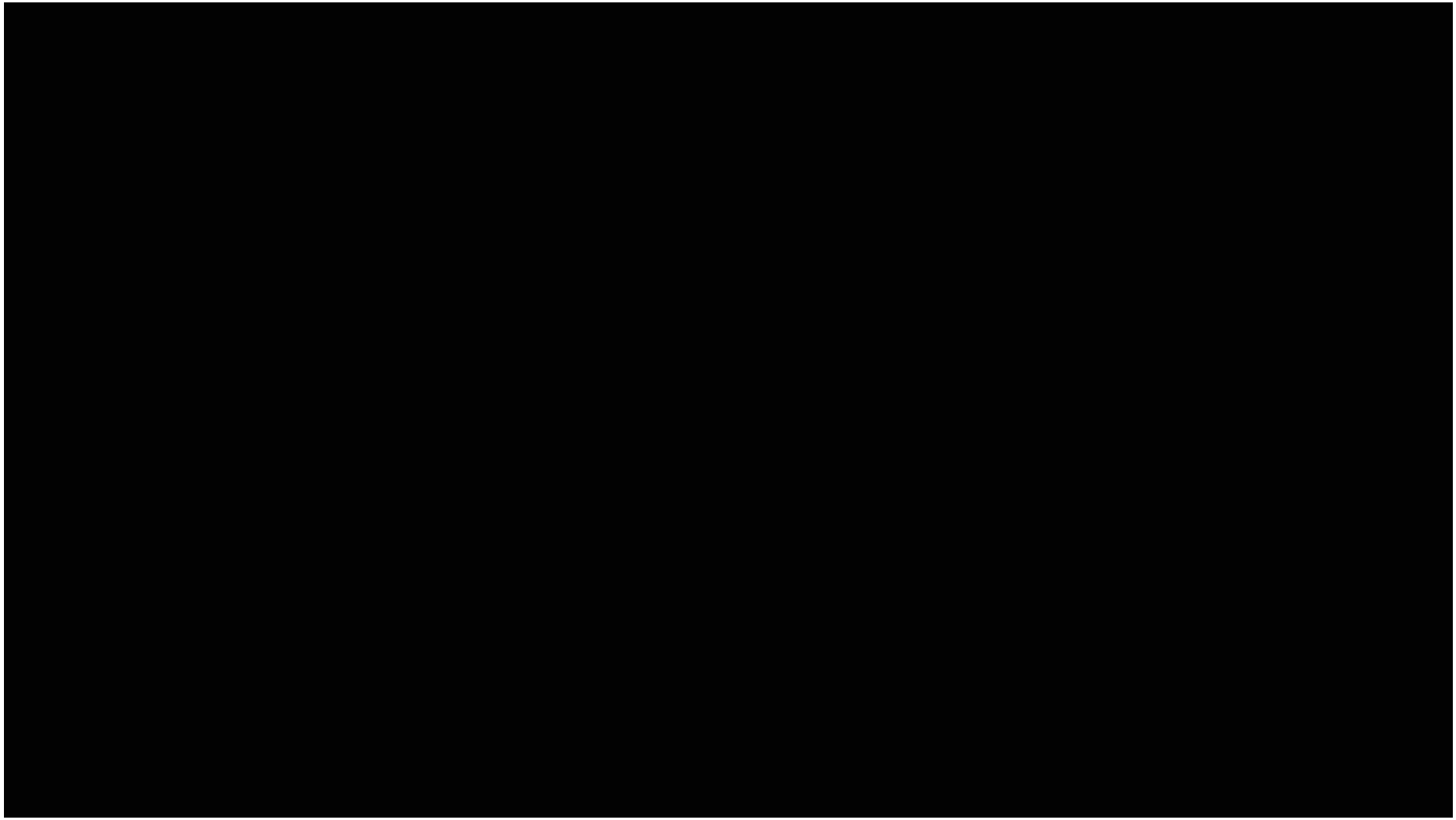
depth image *image RGB*

Kinect v2

- Technologie différente : basée sur temps de vol *(time of flight : ToF)*
- Mesure la phase entre émission lumineuse des LEDs et chaque pixel de la caméra ToF

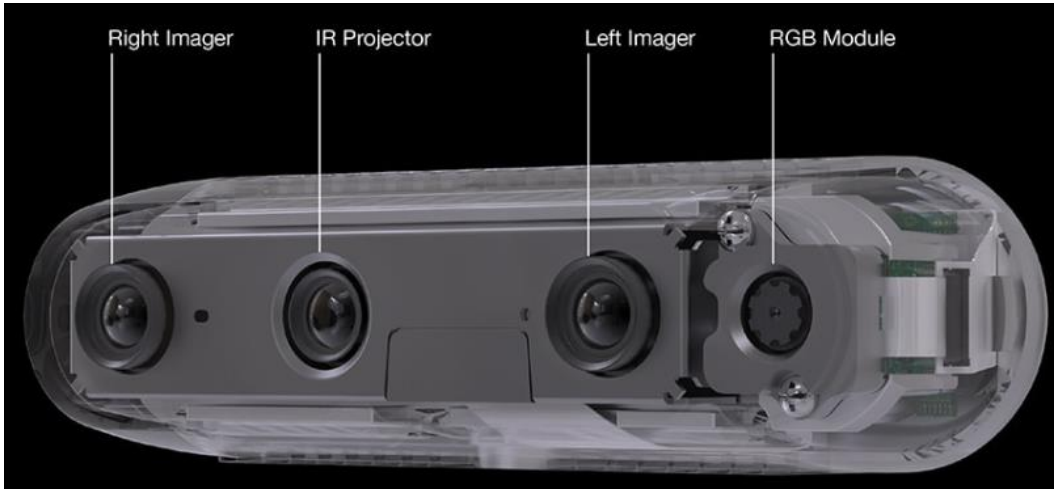


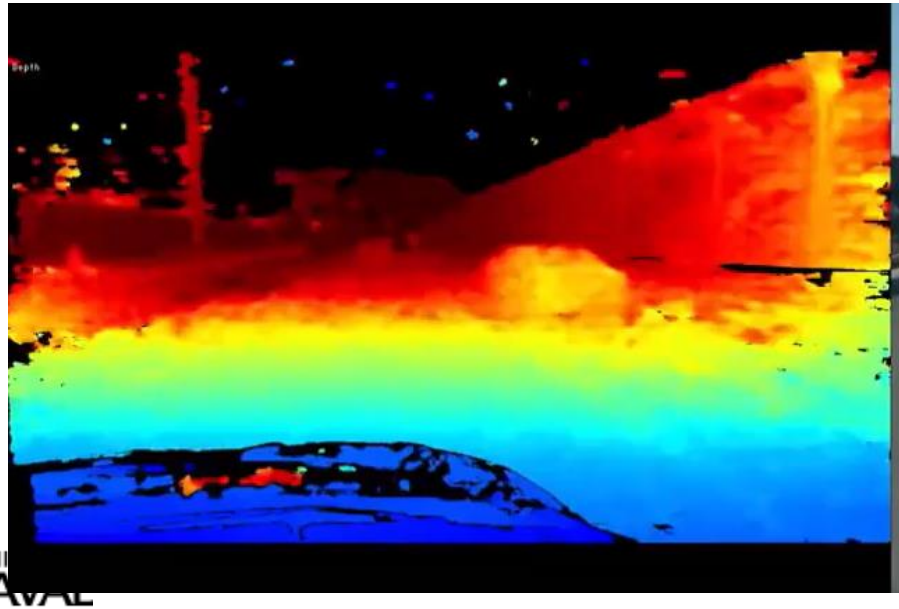
Kinect v2



Intel RealSense

- D435/D435i (IMU)
  - Stéréo normale + active





Autres types de caméras

Caméra omnidirectionnelles

- Caméra standard + miroir convexe = vue 360°

Caméra omnidirectionnelles

- *Ladybug* de *Pointgrey* combine 6 caméras ensemble: vue 360°

même voiture

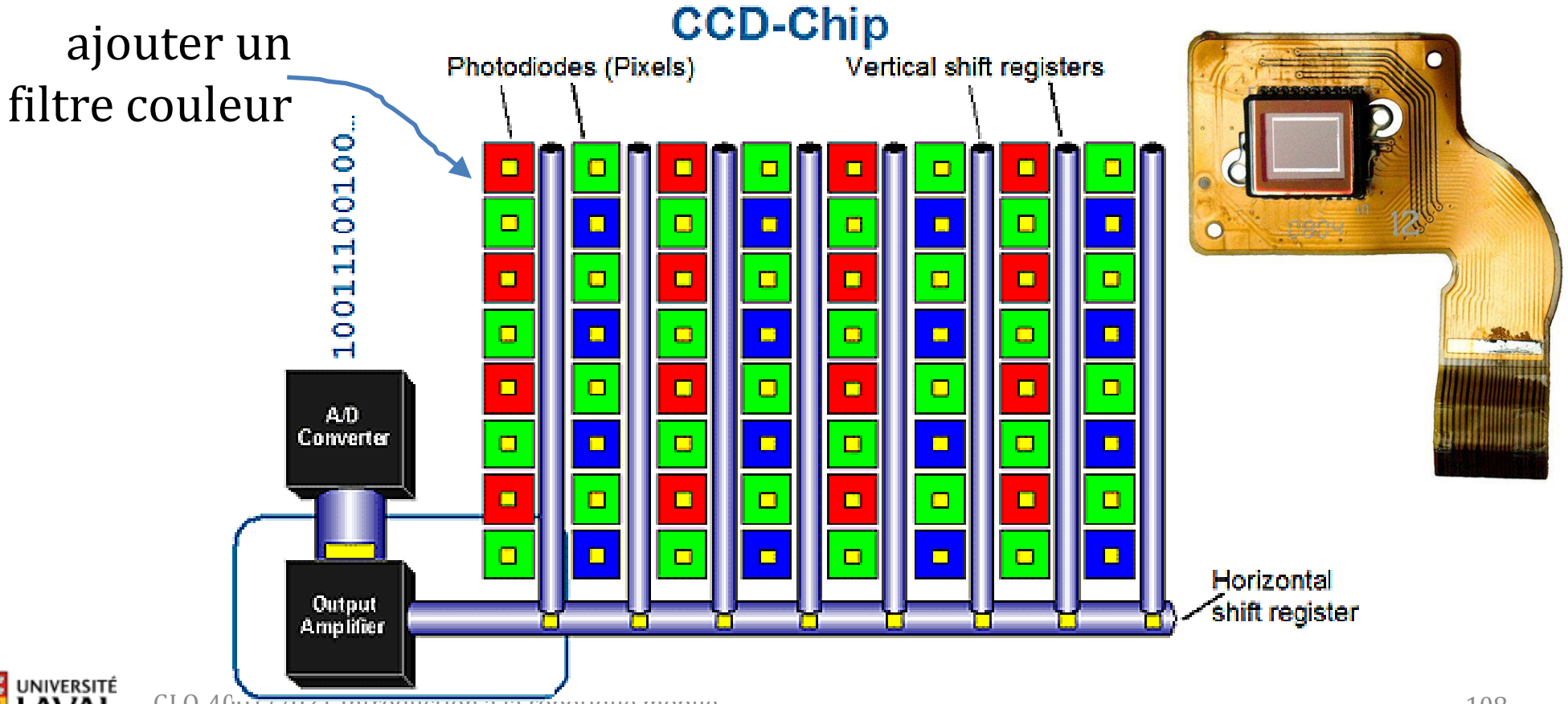
Event camera : DVS128 de iniLabs

- Envoie des messages de changement d'intensité **ON/OFF**
- Temps de réponse autour de 15 microseconde

Traitement d'image

Capteur image CCD + filtre *Bayer*

- Placé sur plan image (en arrière)
- Pixel CCD nu : sensible à toutes les couleurs



Échantillonnage : *aliasing*

- CCD : échantillonnage dans l'espace
- *Aliasing* arrive si $f_{échantillonage} < 2f_{signal}$

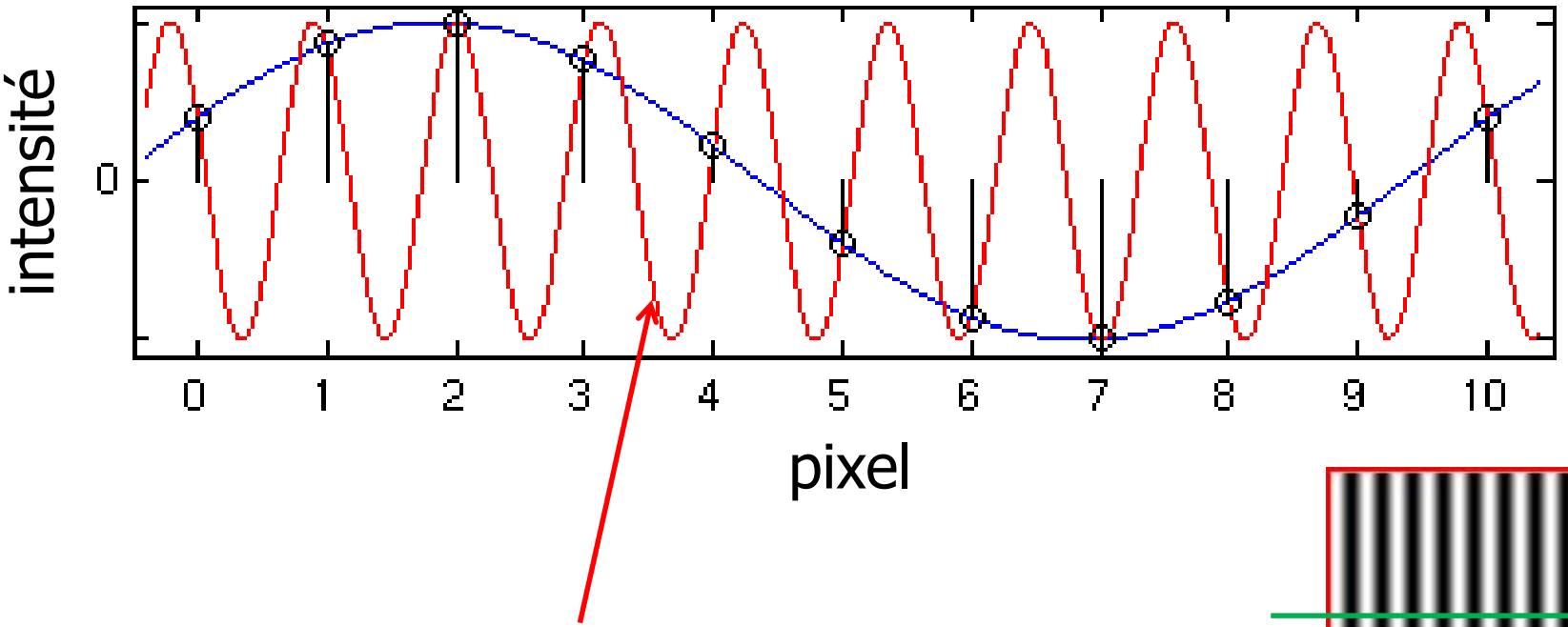
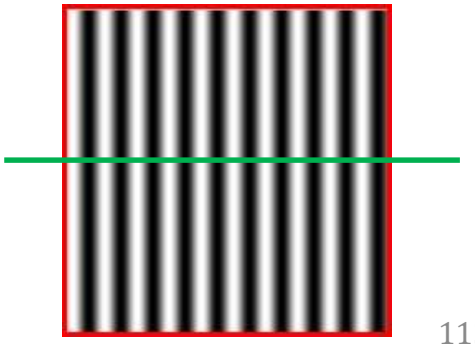
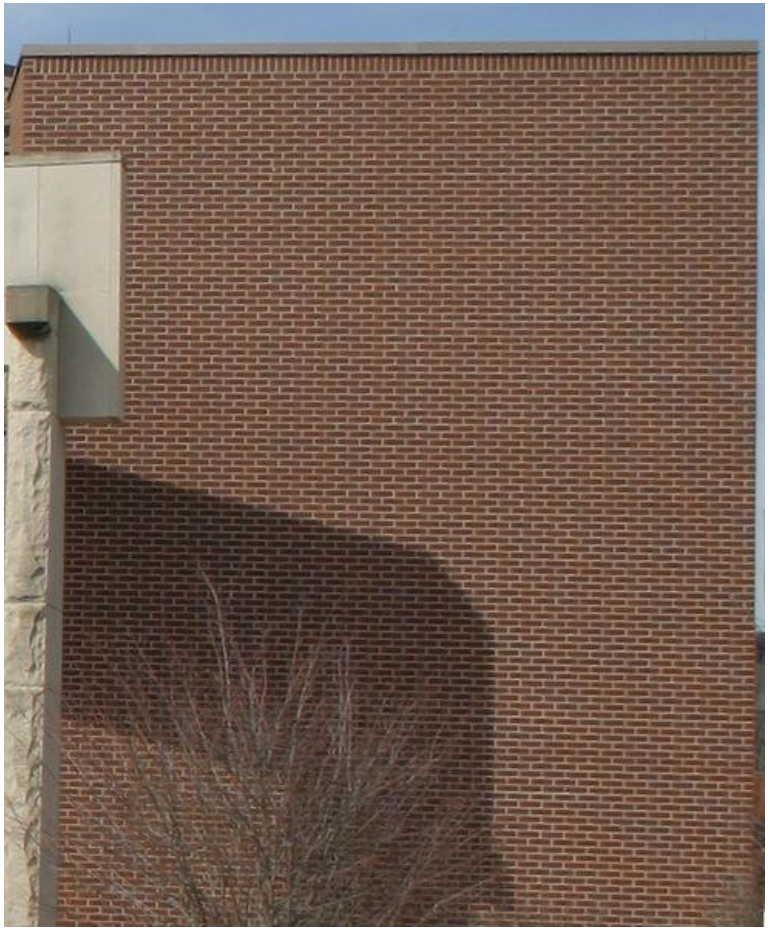


image d'un sinus projeté sur CCD



Aliasing → effet Moiré

- Fréquence spatiale trop élevée

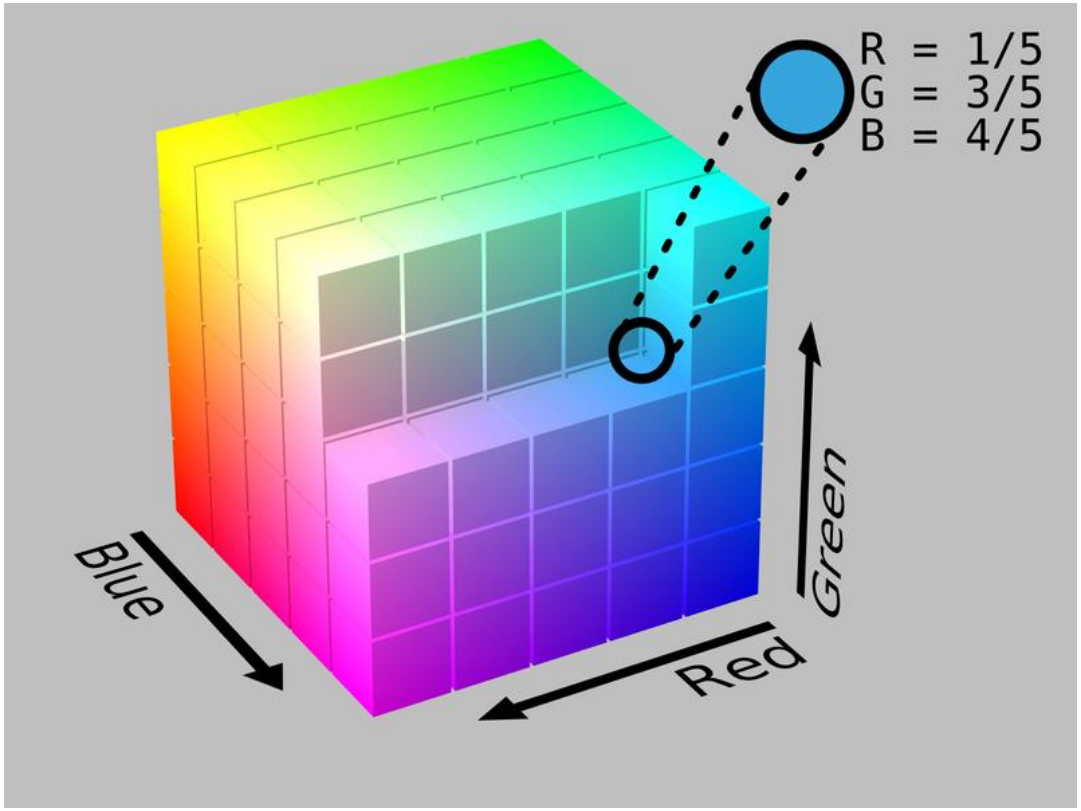


si on prend un pixel sur deux

(il faut filtrer l'image d'origine avec un filtre passe-bas, puis sous-échantillonner)

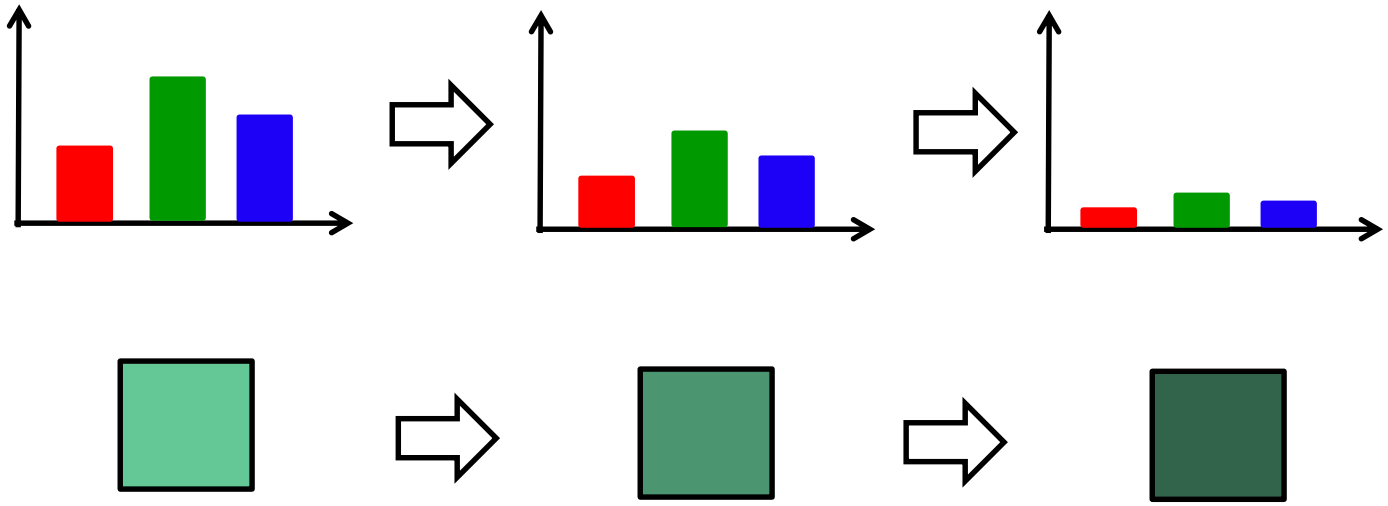
Représentation couleur RGB

- Habitué au RGB : Red Green Blue



Représentation couleur RGB

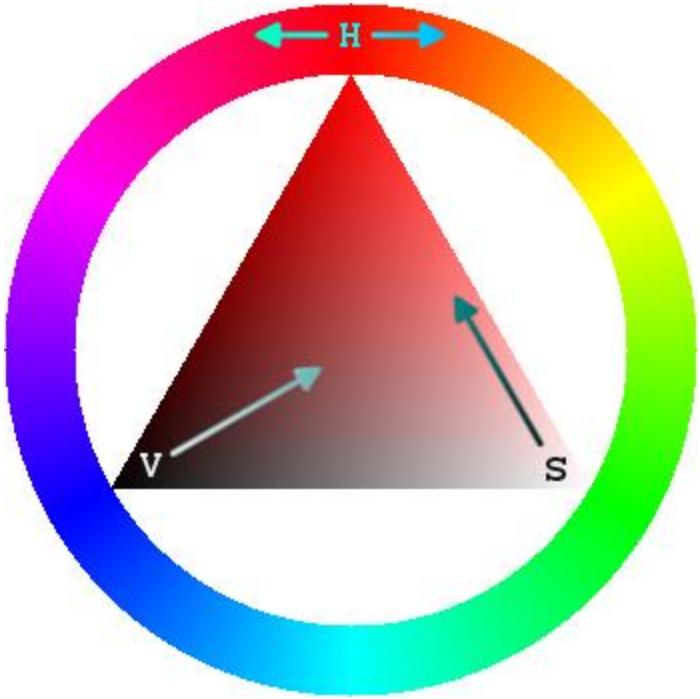
- Que se passe-t-il si l'intensité lumineuse change? (coin ombragé)

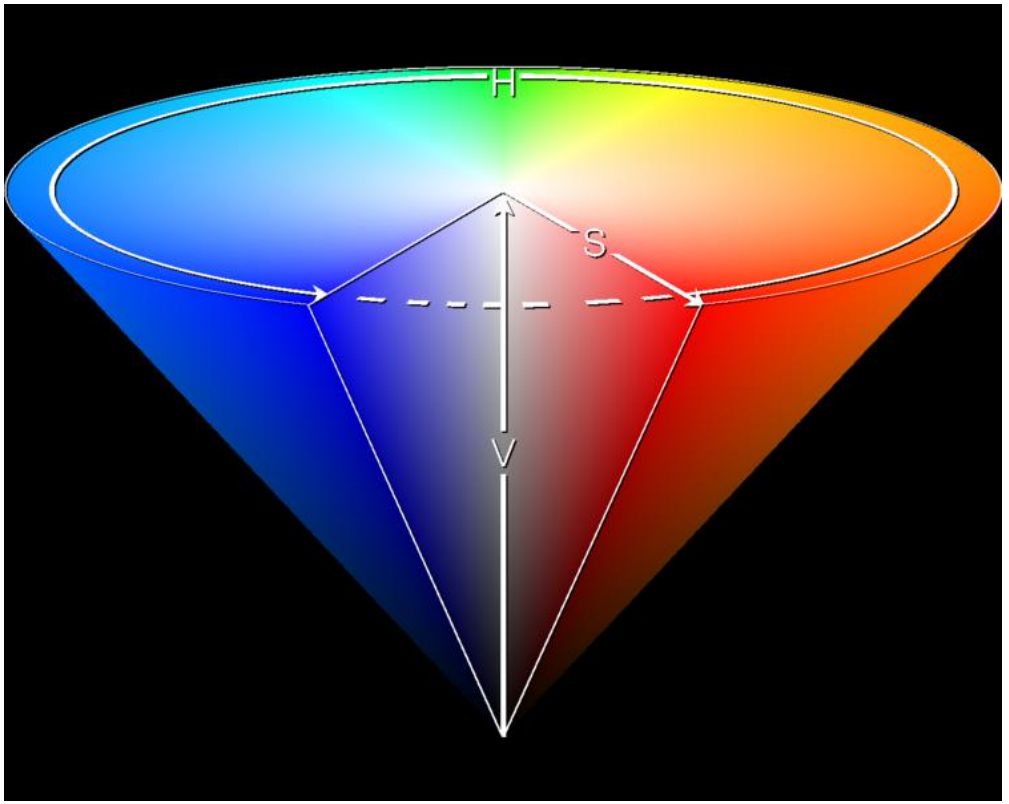


même « teinte », mais plus faible « intensité »

Autre représentation : HSV

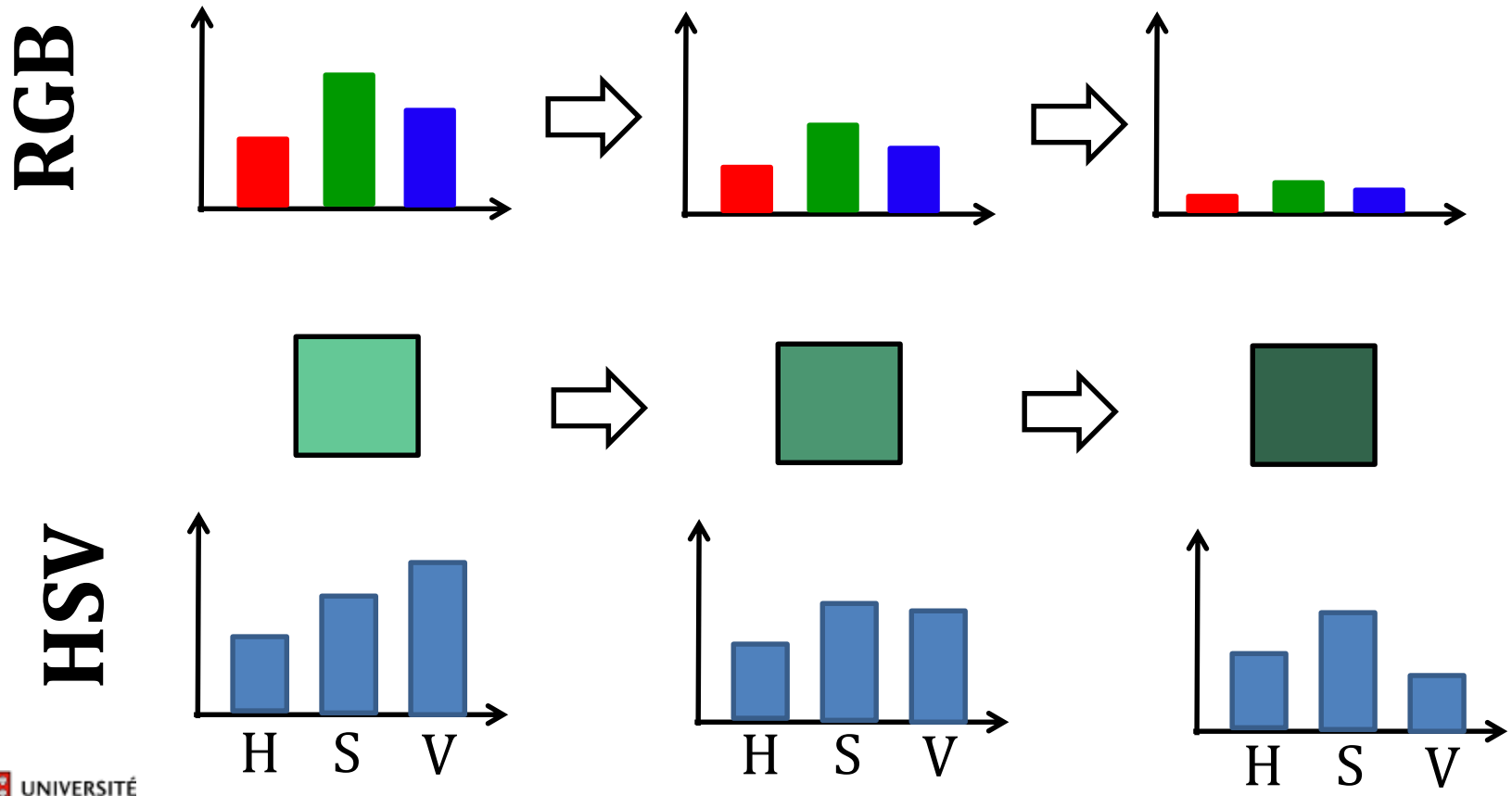
- Hue-Saturation-Value
  - encodage plus près de la perception humaine





HSV vs. RGB

- Que se passe-t-il si l'intensité lumineuse change? (coin ombragé)



Conversion HSV-RGB

RGB → HSV

$$t = \begin{cases} 0, & \text{si } \max = \min \\ (60° \times \frac{g-b}{\max - \min} + 360°) \bmod 360°, & \text{si } \max = r \\ 60° \times \frac{b-r}{\max - \min} + 120°, & \text{si } \max = g \\ 60° \times \frac{r-g}{\max - \min} + 240°, & \text{si } \max = b \end{cases}$$

$$s = \begin{cases} 0, & \text{si } \max = 0 \\ 1 - \frac{\min}{\max}, & \text{sinon} \end{cases}$$

$$v = \max$$

`rgb2hsv` sur matlab

HSV → RGB

$$t_i = \left\lfloor \frac{t}{60} \right\rfloor \mod 6$$

$$f = \frac{t}{60} - t_i$$

$$l = v \times (1 - s)$$

$$m = v \times (1 - f \times s)$$

$$n = v \times (1 - (1 - f) \times s)$$

$$(r, g, b) = \begin{cases} (v, n, l), & \text{si } t_i = 0 \\ (m, v, l), & \text{si } t_i = 1 \\ (l, v, n), & \text{si } t_i = 2 \\ (l, m, v), & \text{si } t_i = 3 \\ (n, l, v), & \text{si } t_i = 4 \\ (v, l, m), & \text{si } t_i = 5 \end{cases}$$

`hsv2rgb` sur matlab

« One line trick for vision »

- Retrouver l'apparence réelle des objets dans les milieux extérieurs (soleil)

(a) Image with shadows

(b) Shadow invariant image

$$F = \log(G) - \alpha \log(R) + \beta \log(B)$$

Dealing with Shadows: Capturing Intrinsic Scene Appearance for Image-based Outdoor Localisation, P. Corke et al., IROS 2013.

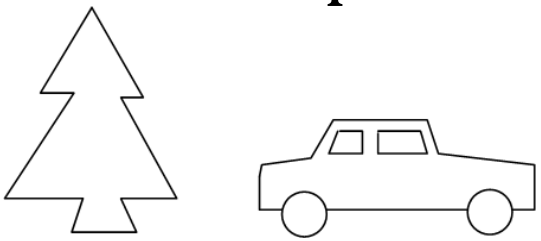
Beyond a Shadow of a Doubt: Place Recognition with Colour-Constant Images, K. MacTavish et al., FSR 2015.

Trouver les sections « intéressantes »

- Traiter l'image complète → temps calcul $$$
- Chercher les points saillants/intéressants
  - équivalent attention visuelle humaine
- Point intéressant:
  - stable (aux changements d'angle/lumière)
  - informatif
  - notion de *distinct du pourtour*

Détection bord (*edge detection*)

- Bords possèdent beaucoup « d'information »



- Certains bords sont plus informatifs que d'autres…

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 4(2):115-147, 1987.

Détection bord (*edge detection*)

- Définition d'un bord : variation spatiale rapide d'intensité lumineuse dans l'image *I*

$$\text{grand gradient } \nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

- Difficile à extraire *parfaitement*
  - Notamment à cause du bruit dans l'image
  - Opération de dérivée **AUGMENTE** l'impact du bruit

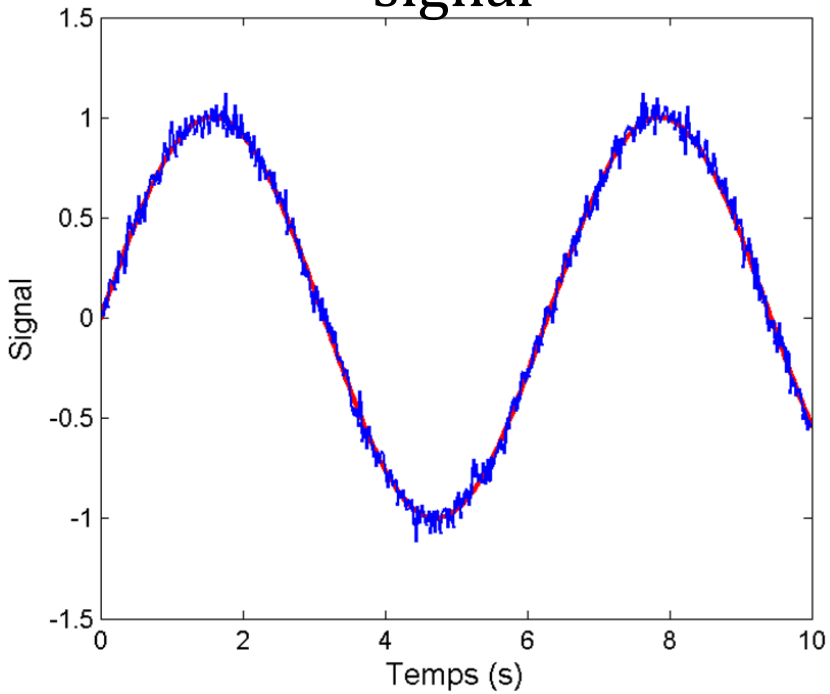
Amplification du bruit par la dérivée

- Signal corrompu avec bruit gaussien $N(0,\sigma^2)$

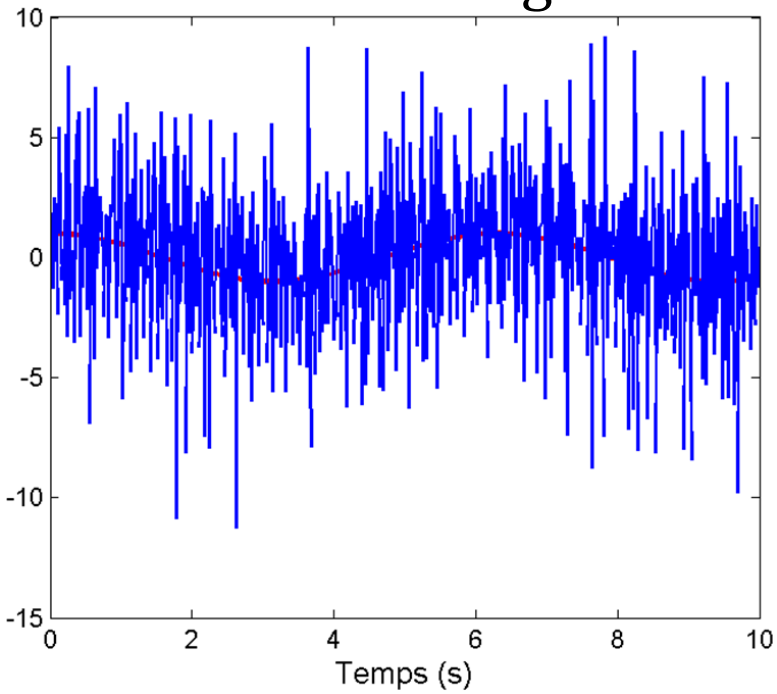
  — signal

  — signal + bruit σ=0.05

signal

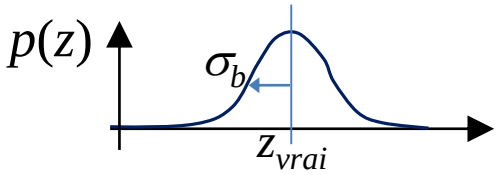


dérivée du signal



Moyenne : réduire le bruit

- Soit la mesure suivante : $z = z_{vrai} + \varepsilon_b, \quad \varepsilon_b \sim N(0, \sigma_b^2)$



constante

ε_b pigé dans une distribution gaussienne non-biaisée

- Si je prends **1** mesure, $\sigma_z^2 = \sigma_b^2$
- Si bruit est une variable aléatoire *iid*

__iid__ : indépendantes et identiquement distribuées

  - valeur du bruit à *t+1* est indépendante valeur à *t*
- Prendre la moyenne de **2** mesures $Z_{moy} = \frac{1}{2} Z_1 + \frac{1}{2} Z_2$

$$\mathrm{Var}(aX + bY) = a^2 \mathrm{Var}(X) + b^2 \mathrm{Var}(Y) + 2ab\,\mathrm{Cov}(X, Y)$$

$$\sigma_{moy}^2 = \mathrm{var}\{Z_{moy}\} = \mathrm{var}\{\frac{1}{2} Z_1 + \frac{1}{2} Z_2\} = (\frac{1}{2})^2 \mathrm{var}\{Z_1\} + (\frac{1}{2})^2 \mathrm{var}\{Z_2\} = 2(\frac{1}{2})^2 \sigma_b^2 = \frac{1}{2}\sigma_b^2$$

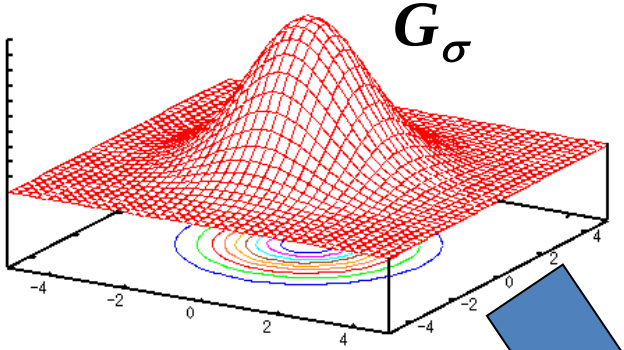
- De façon générale, pour ***N*** mesures moyennées

$$\sigma_{moy}^2 = \frac{1}{N}\sigma_b^2 \quad \Rightarrow \quad \sigma_{moy} = \frac{1}{\sqrt{N}}\sigma_b$$

Moyenne image : flou gaussien G_σ

convolution

*



=

flou volontaire pour réduire l'impact du bruit

...vient faire la ***moyenne pondérée*** avec pixels voisins...

pixel loin a moins d'influence

Gaussian Window

w1 .004	w2 .015	w3 .026	w4 015	w5 .004
w6 015	w7 .059	w8 .095	w9 .059	w10 015
w11 .026	w12 .095	w13 .15	w14 .095	w15 .026
w16 015	w17 .059	w18 .095	w19 .059	w20 015
w21 .004	w22 015	w23 .026	w24 015	w25 .004

Sobel *edge detector,* avec bruit

approxime le gradient

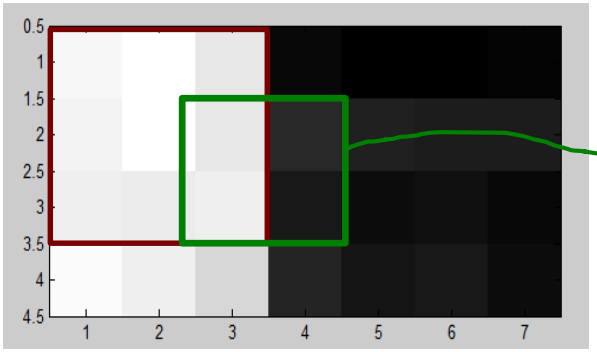
$$\Delta_1 = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

$$Sobel = \sqrt{\Delta_1^{\,2} + \Delta_2^{\,2}}$$

207	210	195	63	57	56	59
204	212	197	82	76	74	75
202	198	202	72	65	67	63
209	201	187	78	69	71	64

$\Delta_1(x,y)$

	-26	-533	-517	-28	
	-29	-505	-513	-25	

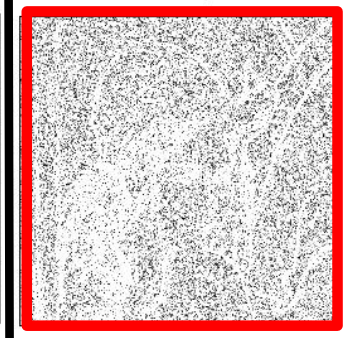


bord == grande valeur $|\Delta_i|$

Sobel

image originale

image bruitée



amplification du bruit

UNIVERSITÉ LAVAL

Filtrage flou : « lissage » du bruit

$I * G_\sigma$

diamètre gaussienne G_σ

gaussienne plus large → image plus floue

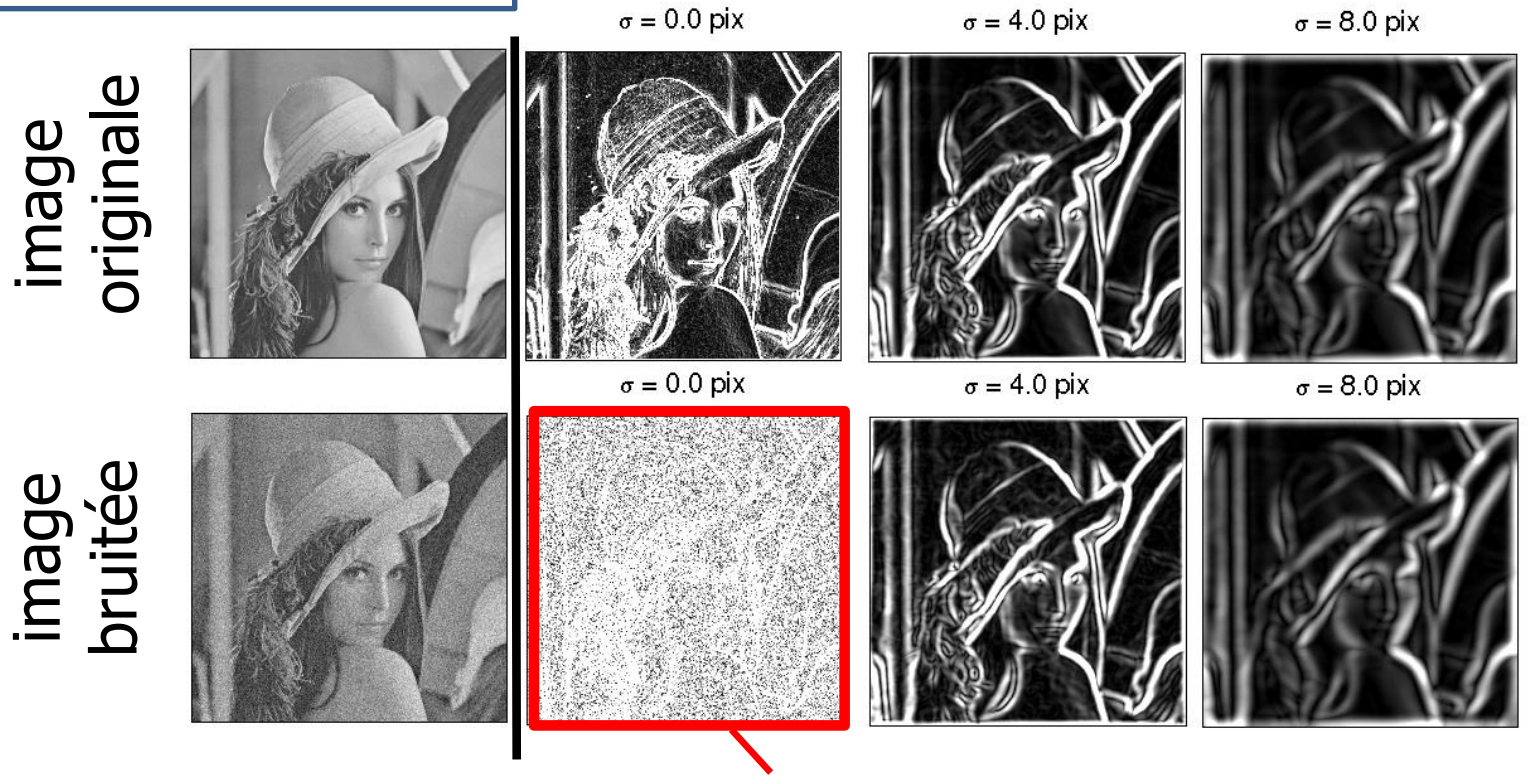
(convolution est comme une somme pondérée)

Sobel *edge detector,* avec bruit

approxime le gradient

$$\Delta_1 = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

$$Sobel = \sqrt{\Delta_1^{\,2} + \Delta_1^{\,2}}$$



Repères visuels naturels

Repères naturels

- Pas tous les endroits dans une images qui sont utiles pour la localisation
- Détecter des endroits « saillants » dans l'image, ex. coins : **keypoint**
- Calculer une signature visuelle autour de ce point (**descripteur**)
- *keypoint* + descripteur = *feature*
- Pourquoi ne pas calculer une signature autour de chaque pixel? Temps de calcul!

Détecteurs de coins
(keypoint detectors)

Propriétés d'un *keypoint*

- Diffère de ses voisins en terme de texture, couleur et/ou intensité (***distinctiveness***)
- Détection répétable (***repeatability***) d'une image à l'autre malgré des changements
  - de points de vue (rotation/translation/affine[1])
  - d'illumination
- Localisé (occupe espace restreint)
- Rapide à calculer (on parcourt toute l'image)
- En trouve de nombreux dans les images (+100)

[1] *transformation qui préserve les points, droites, plans et parallélisme (rotation, shear, translation, etc.)*

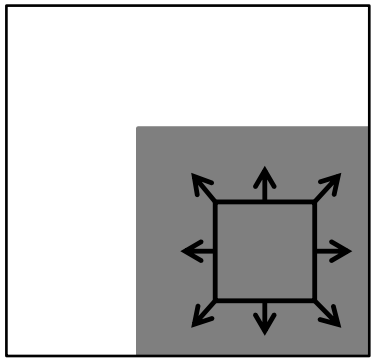
Exemple : *Moravec interest operator*

- Utilise Sum-of-Squared Difference (SSD) comme mesure de similarité entre deux patches I_a et I_b :

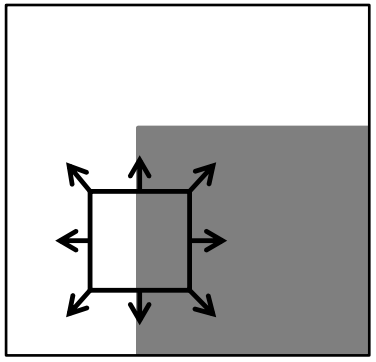
$$SSD(I_a, I_b) = \sum_{i,j \in patch} \left(I_a(i,j) - I_b(i,j)\right)^2$$

- Cherche un endroit où le SSD par rapport aux patches voisines est localement maximal :

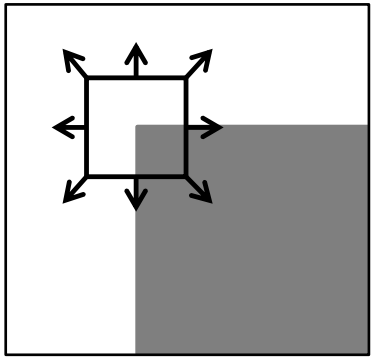
pas



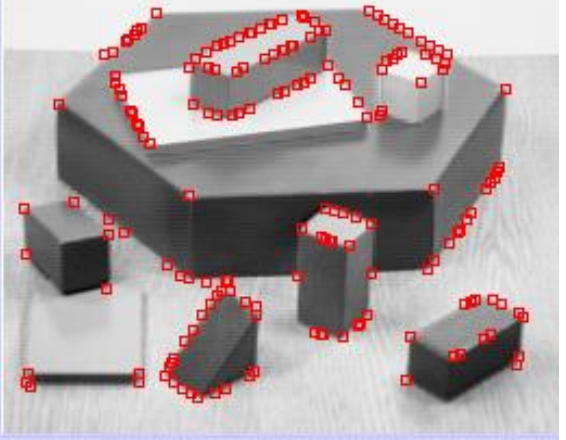
un peu



beaucoup



exemples de coins

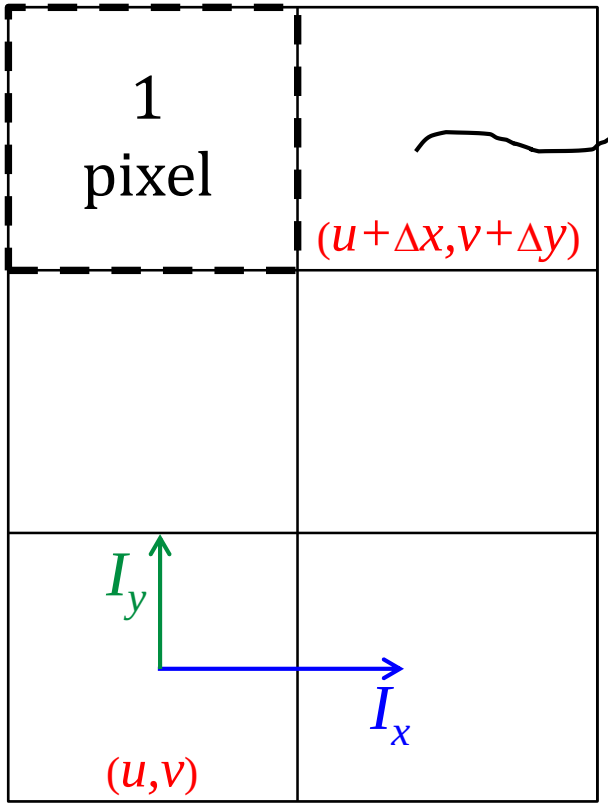


UNIVERSITÉ LAVAL

Harris corner detector

- Approximation par Taylor :
  - gradients (calculé avec Sobel) $I_x(i,j) = \frac{\partial}{\partial x} I(i,j), \quad I_y(i,j) = \frac{\partial}{\partial y} I(i,j)$

Note : il est un peu fâcheux que les livres utilisent I_x, I_y comme un gradient, car ils rappellent plus une intensité (image I)



$$I(u+\Delta x, v+\Delta y) \approx I(u,v) + I_x(u,v)\Delta x + I_y(u,v)\Delta y$$

Basé sur le changement d'apparence SSD.

Pour une paire de pixels :

$$\left(I(u+\Delta x, v+\Delta y) - I(u,v)\right)^2 \approx \left(I(u,v) + I_x(u,v)\Delta x + I_y(u,v)\Delta y - I(u,v)\right)^2$$

$$\approx \left(I_x(u,v)\Delta x + I_y(u,v)\Delta y\right)^2$$

$$\approx I_x(u,v)^2 \Delta x^2 + 2I_x(u,v)I_y(u,v)\Delta x\Delta y + I_y(u,v)^2 \Delta y^2$$

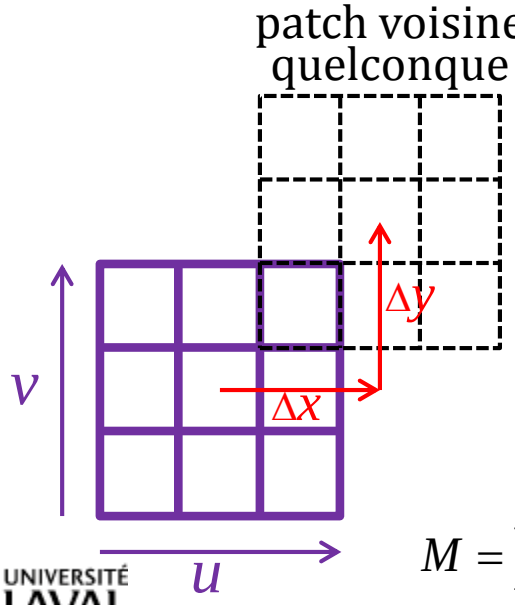
$$\approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

Harris corner detector

- Comme pour Moravec, on regarde la distribution des SSD avec les patches voisines

SSD 1 paire pixel : $\left(I(u+\Delta x, v+\Delta y) - I(u,v)\right)^2 \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$

Le SSD <u>d'une patch au complet</u> décalée par Δx, Δy est :



$$\underset{patch}{\mathrm{SSD}}(\Delta x, \Delta y) \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \left(\sum_{u,v} \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v) \end{bmatrix} \right) \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

$$\underset{patch}{\mathrm{SSD}}(\Delta x, \Delta y) \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} M \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

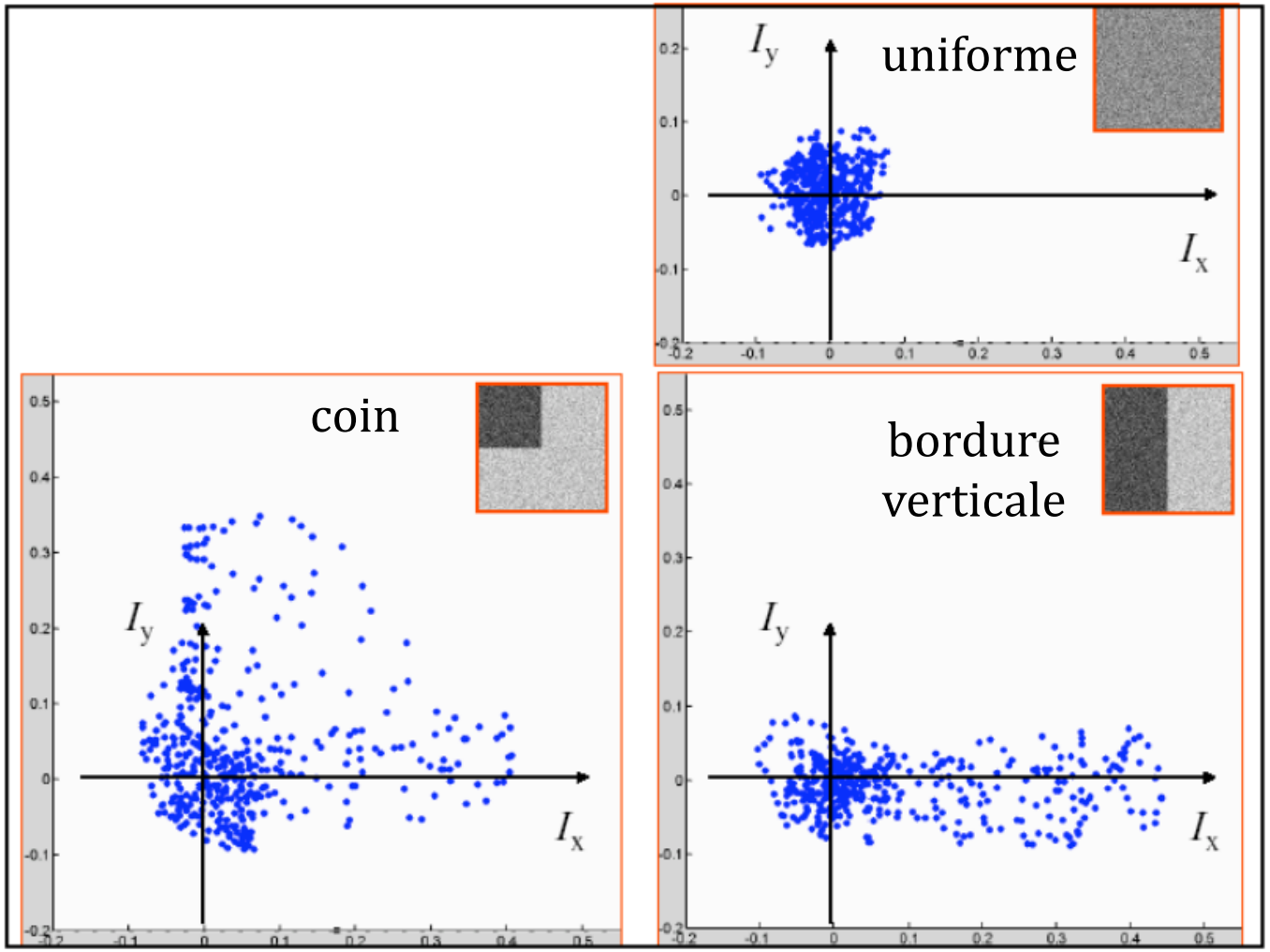
$$M = \sum_{u,v} \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v) \end{bmatrix}$$

Nous donne une idée de la variabilité locale d'une patch

UNIVERSITÉ LAVAL

Harris corner detector

- Distribution des dérivées I_x, I_y



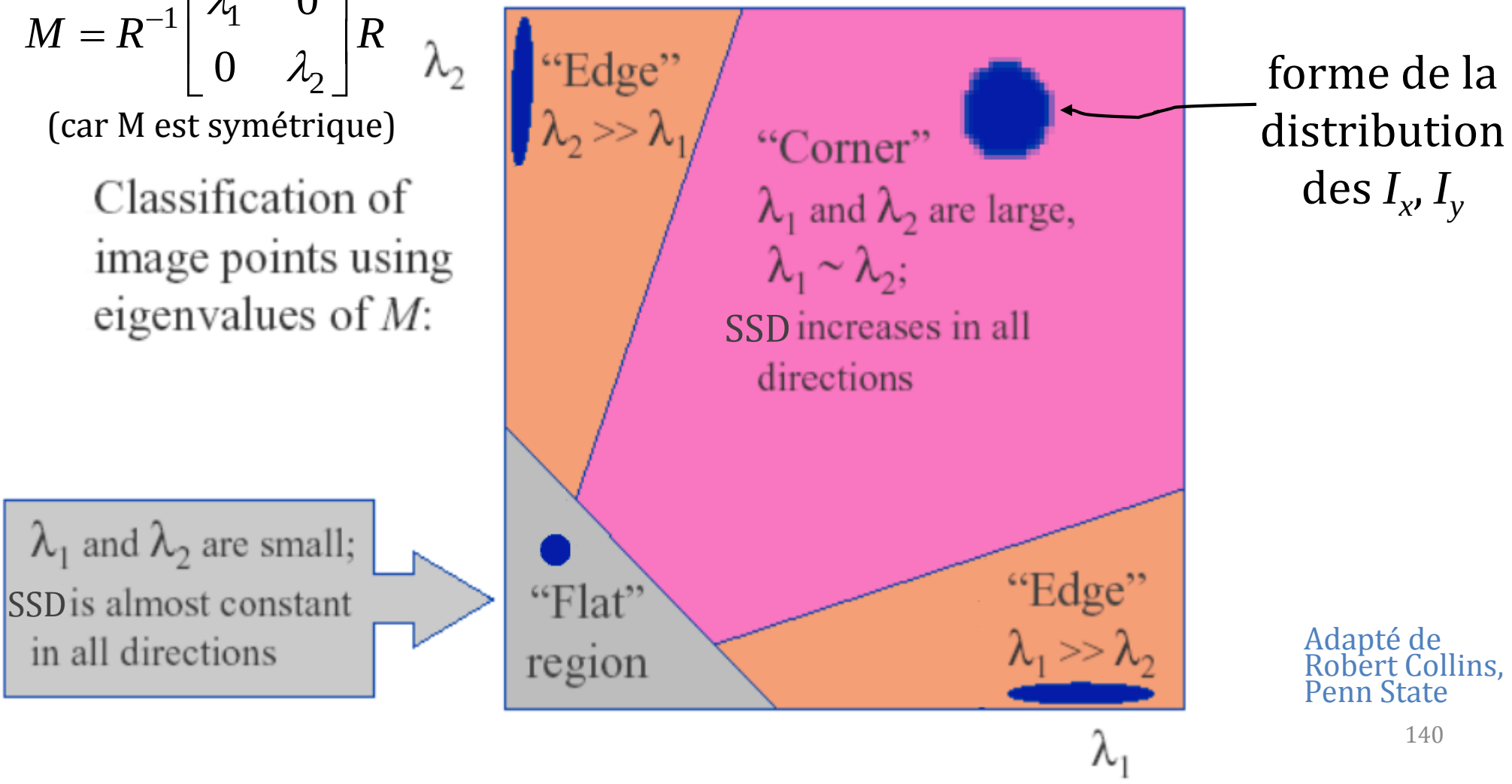
Adapté de Robert Collins, Penn State

Harris corner detector

- Distribution des valeurs propres λ_1, λ_2 de M

$$M = R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$$

(car M est symétrique)



Adapté de Robert Collins, Penn State

HarrisExample.m

Harris corner detection : Algorithme

Adapté de Robert Collins, Penn State

```
SobelX = [-1 0 1; -2 0 2; -1 0 1];
SobelY = [1 2 1; 0 0 0; -1 -2 -1];
Ix = conv2(SobelX,I);
Iy = conv2(SobelY,I);

Ix2 = Ix.^2;
Iy2 = Iy.^2;
Ixy = Ix.*Iy;

G = fspecial('gaussian', [5 5], 1);
Sx2 = conv2(Ix2,G);
Sy2 = conv2(Iy2,G);
Sxy = conv2(Ixy,G);
```

```
G=
   0.0030   0.0133   0.0219   0.0133   0.0030
   0.0133   0.0596   0.0983   0.0596   0.0133
   0.0219   0.0983   0.1621   0.0983   0.0219
   0.0133   0.0596   0.0983   0.0596   0.0133
   0.0030   0.0133   0.0219   0.0133   0.0030
```

```
k = 0.1; % Parametre kappa
R = Sx2.*Sy2-Sxy.^2-k*((Sx2+Sy2).^2);
```

```
[CornerX CornerY] = find(R>10000000);
```

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x.I_x \quad I_{y2} = I_y.I_y \quad I_{xy} = I_x.I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma\prime} * I_{x2} \quad S_{y2} = G_{\sigma\prime} * I_{y2} \quad S_{xy} = G_{\sigma\prime} * I_{xy}$$

une convolution peut être vue comme une somme pondérée

4. Define at each pixel (x, y) the matrix

$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

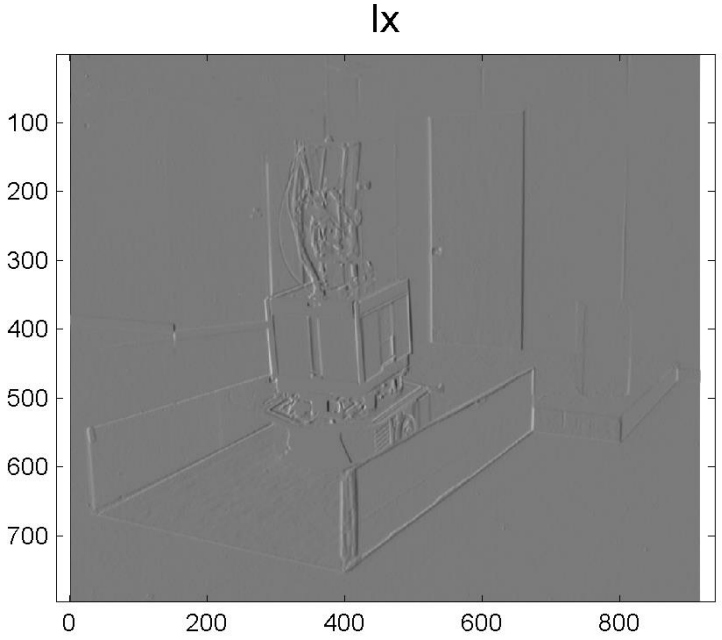
5. Compute the response of the detector at each pixel

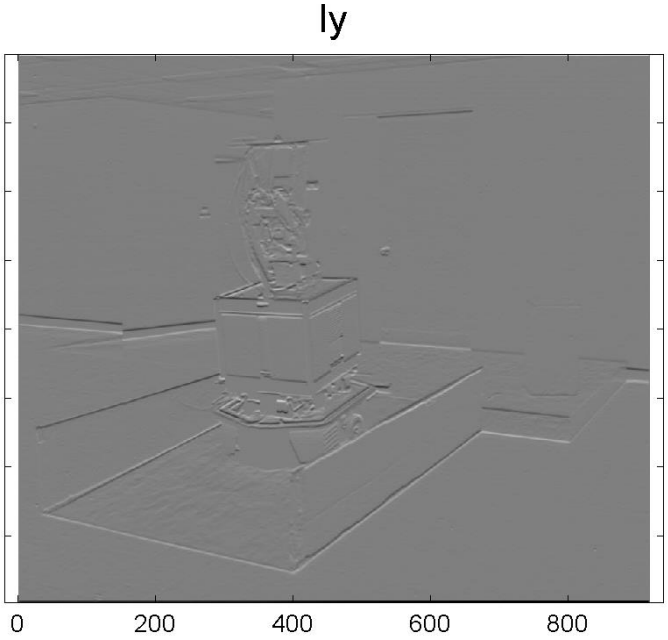
$$R = Det(H) - k(Trace(H))^2$$

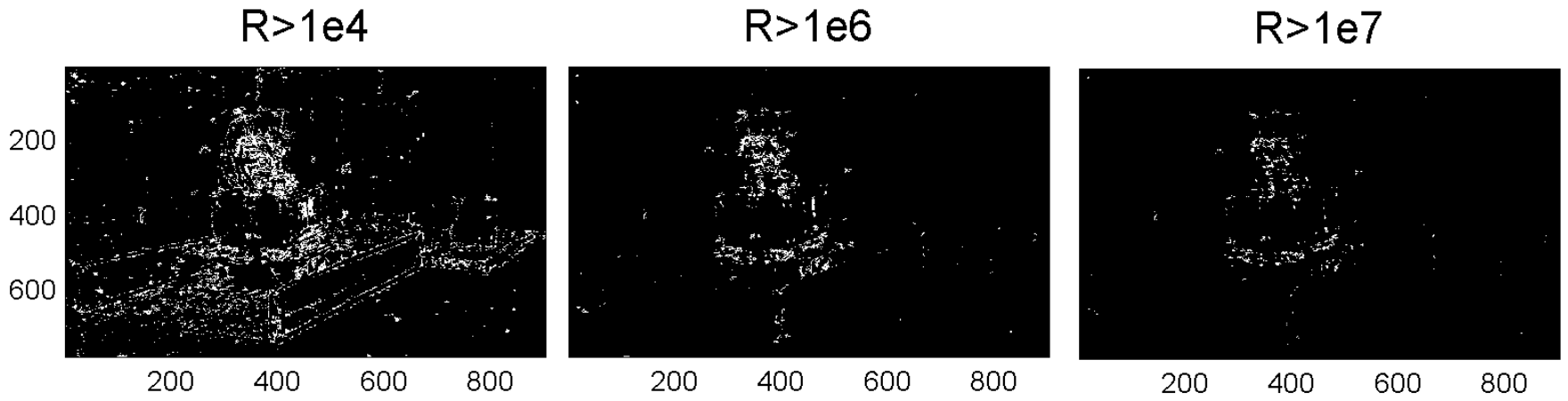
critère relié aux valeurs propres (rapide à calculer)

6. Threshold on value of R. Compute nonmax suppression.

Résultats Harris

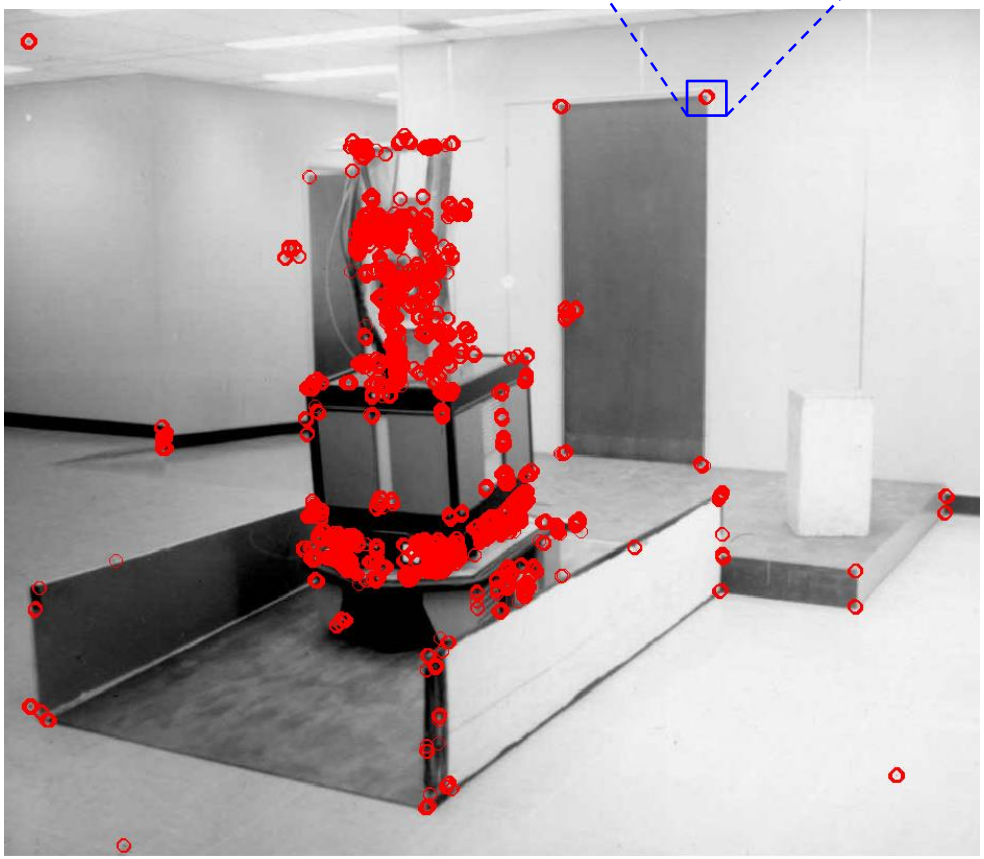






Résultats Harris (sans nonmax suppr.)

La suppression des non-maxima locaux retirerait ces doublons pour n'en garder qu'un seul

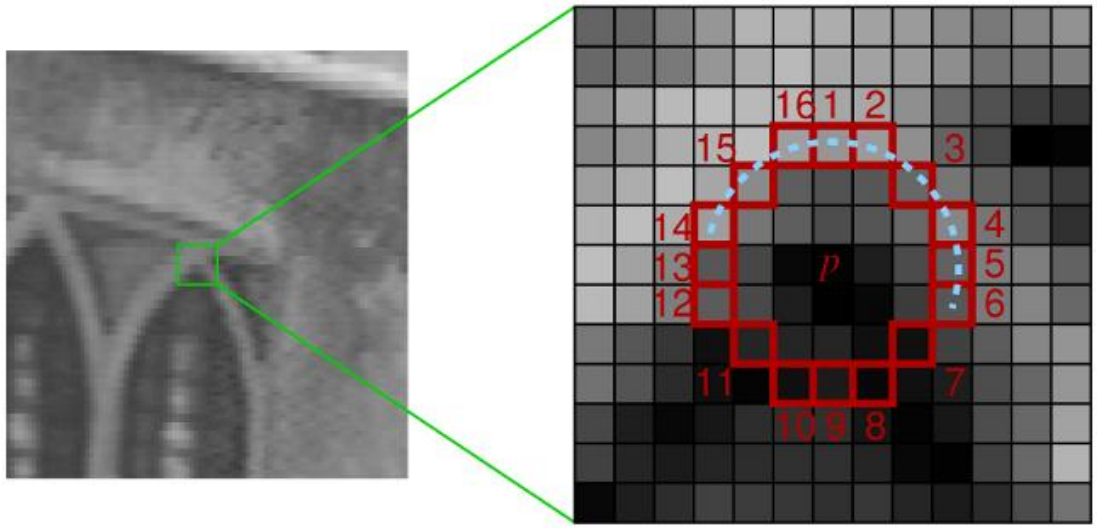


FAST : Features from Accelerated Segment Test

- Autre détecteur de coins
- Cherche un **arc continu** de N ou + pixels
  – un peu plus intense que le point central p : $I(x)>(I(p)+t)$
  ou
  – un peu moins intense que le point central p : $I(x)<(I(p)-t)$

point sur l'arc

t: threshold

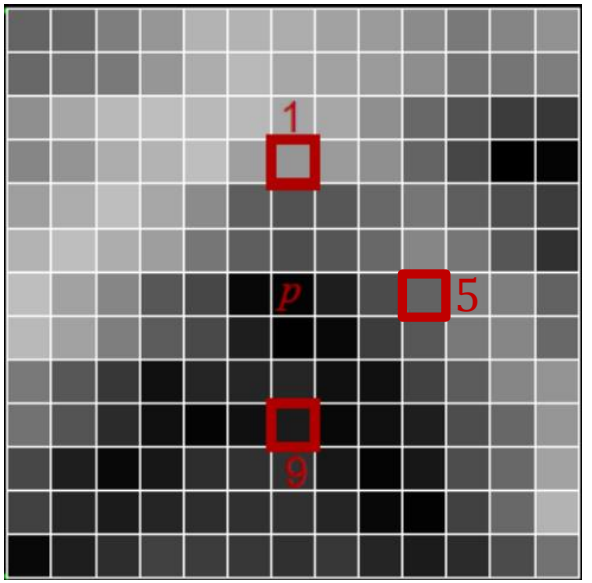


© Edward Rosten

E. Rosten, T. Drummond, Machine Learning for High-Speed Corner Detection, ECCV 2006.
http://www.edwardrosten.com/work/fast.html

Détecteur FAST

- Pour $N \geq 12$, sur rayon de 3 pixels, rapidité!
  - on test si 1,9 respecte le critère
  - puis 5…
  - puis 13.
- Si on a trois points qui respectent le critère, on test tous les 12 autres points restants
- Suppression des non-maxima locaux du score $V = \max \begin{cases} \sum(\text{pixel values} - p) \text{ if } (\text{value} - p) > t \\ \sum(p - \text{pixel values}) \text{ if } (p - \text{value}) > t \end{cases}$



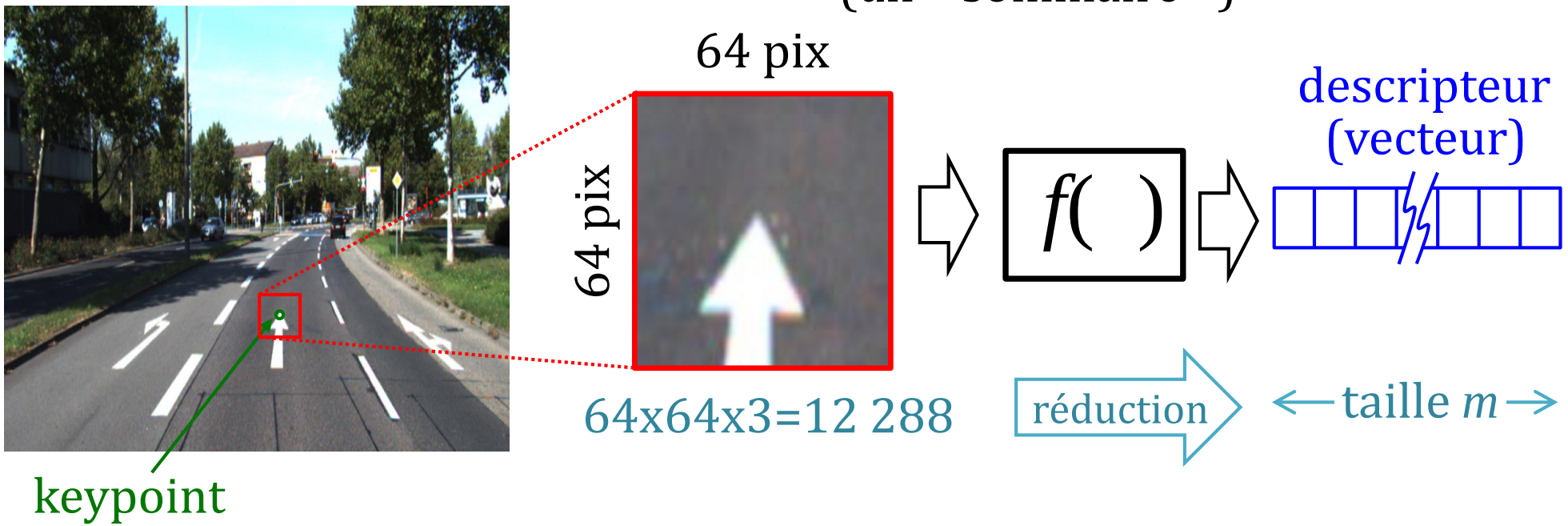
(somme des différences absolues entre le pixel p et tous les pixels de l'arc continu)

FAST est 20x plus rapide que Harris

Repères naturels : Descripteurs

Descripteurs

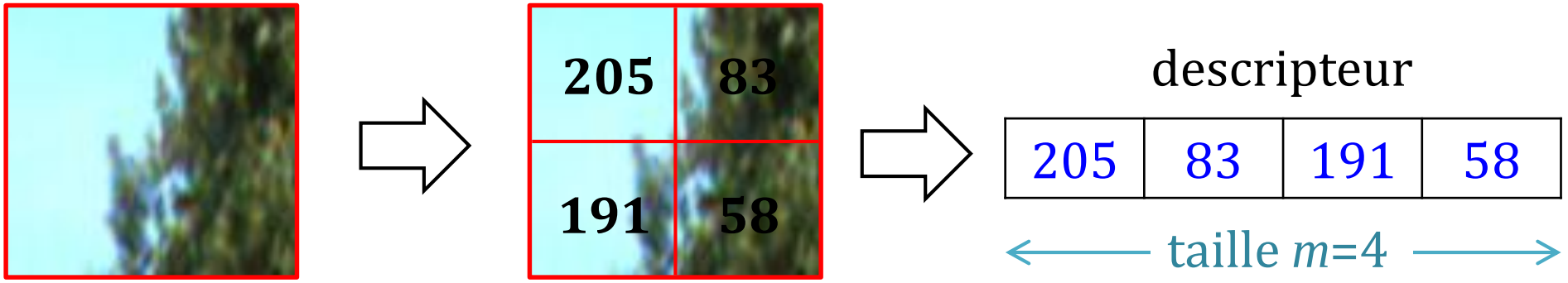
- 1ère étape : identifier les keypoints
- 2ème étape : leur attribuer une signature=descripteur
  (un « sommaire »)



- Au final, *feature* = position (u,v) dans l'image + descripteur

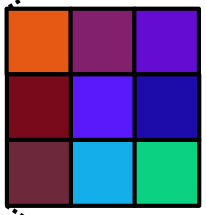
Exemple simplifié

- Vecteur des moyenne d'intensités 2x2

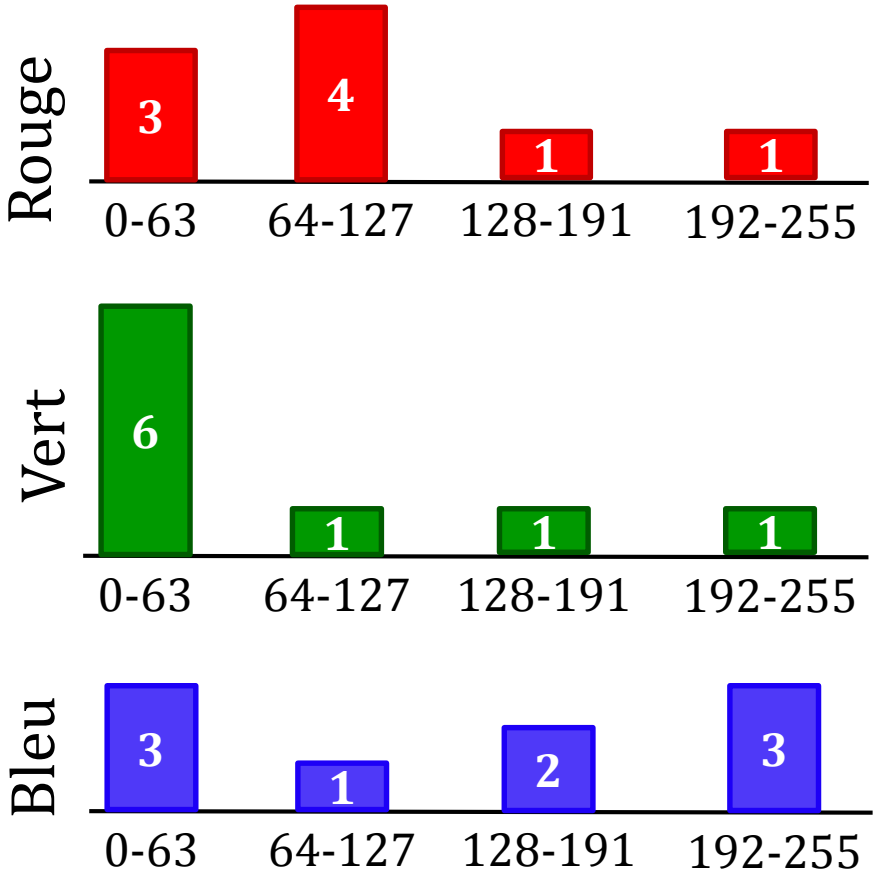


Ex. 2 : Histogramme de couleurs

Image de 3x3 pixel



R: 230 G: 89 B: 21	R: 130 G: 32 B: 110	R: 100 G: 12 B: 210
R: 120 G: 10 B: 27	R: 90 G: 25 B: 250	R: 30 G: 12 B: 170
R: 110 G: 40 B: 60	R: 20 G: 175 B: 235	R: 10 G: 210 B: 130



taille m=12

Descripteur:

3	4	1	1	6	1	1	1	3	1	2	3

Descripteurs : propriétés désirables

- **Compact**
  - dimension *m* de taille plus faible que la patch
  - ex: patch de 64x64 pixels → vecteur taille *m*=128
- **Rapide à calculer**
  - ex: 1000 desc./image x 30 image/s = 30 000 desc/s
- **Distinctif**
- **Répétable**
  - robuste aux changements de points de vue (rotation, translation, échelle) et d'illumination

Appariement par force brute

- Une fonction de distance $s(f_i, f_j)$ qui compare les deux descripteurs f_i et f_j

$$s(f_i, f_j) = \| f_i - f_j \|_2 = \sqrt{\sum_{a=1}^{m} (f_i^a - f_j^a)^2}$$

FLANN: Fast Library for Approximate Nearest Neighbors

KD-trees

distance cosine $s(f_i, f_j) = f_i \bullet f_j$

- Pour tous descripteurs f_i dans l'image 1
  - calculer $s(f_i, f_j)$ pour tous descripteurs f_j dans image 2
  - conserver le descripteur f_j le plus proche* de f_i

*dans l'espace des descripteurs $\mathbb{R}^m$

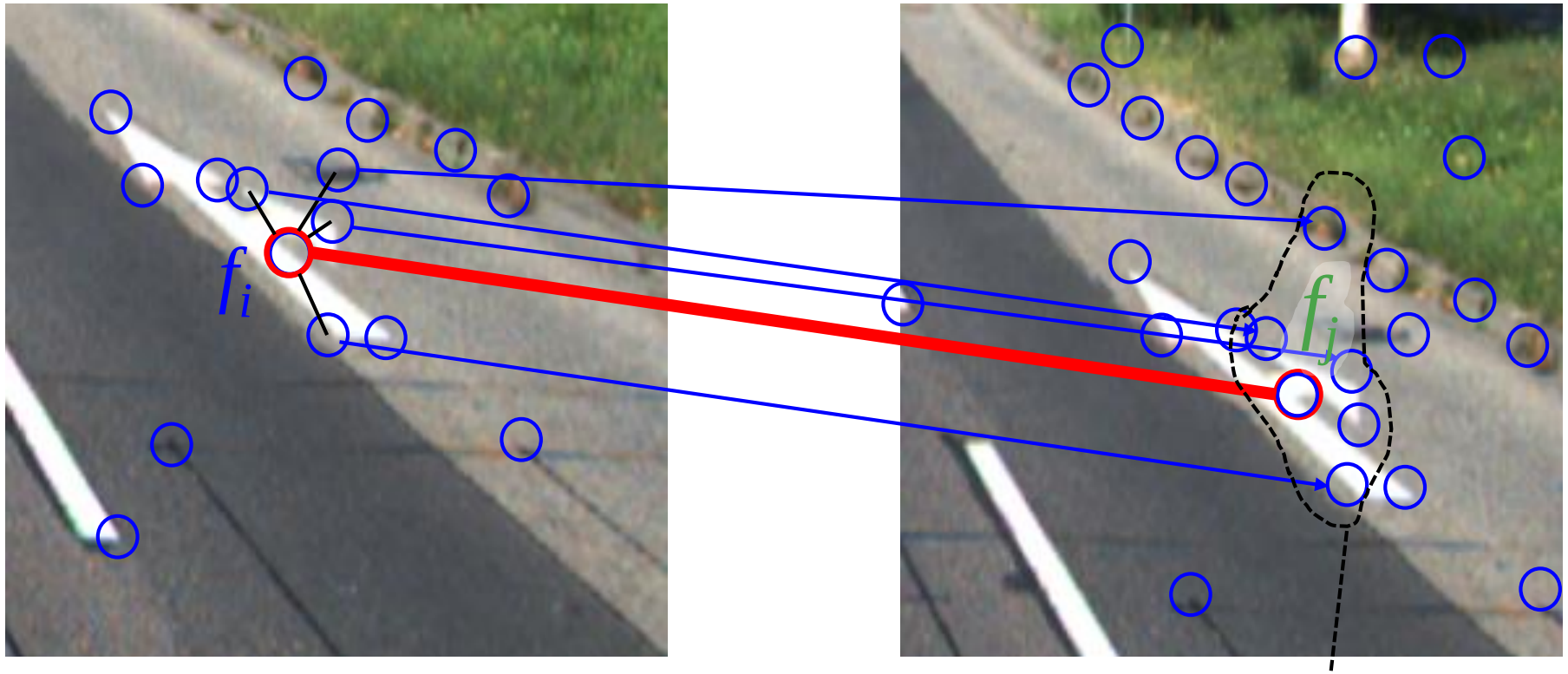
Vérifications (1)

- Seuil maximal sur la distance $s(f_i, f_j)$

- **Test de Lowe**:
  - distance s_1 du plus similaire selon $s(f_i, f_j)$
  - distance s_2 du **deuxième** plus similaire selon $s(f_i, f_j)$
  - ratio des distances s_1 / s_2
  - conserve si ce ratio est inférieur à p (typique 0.6-0.8)

Vérifications (2)

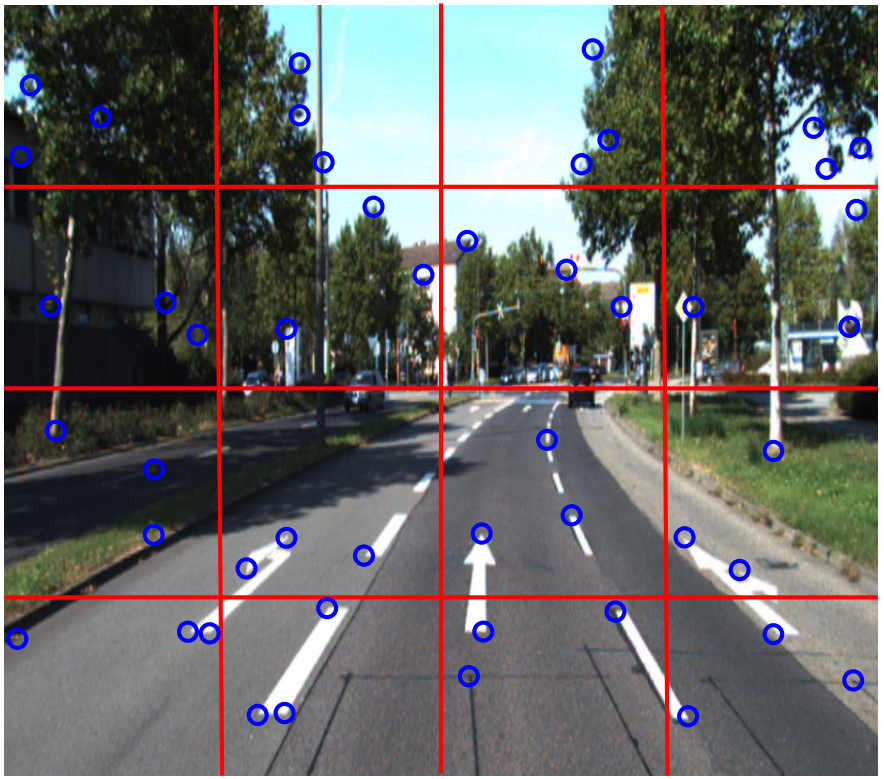
- **Vérification géométrique**



Répartition sur l'image

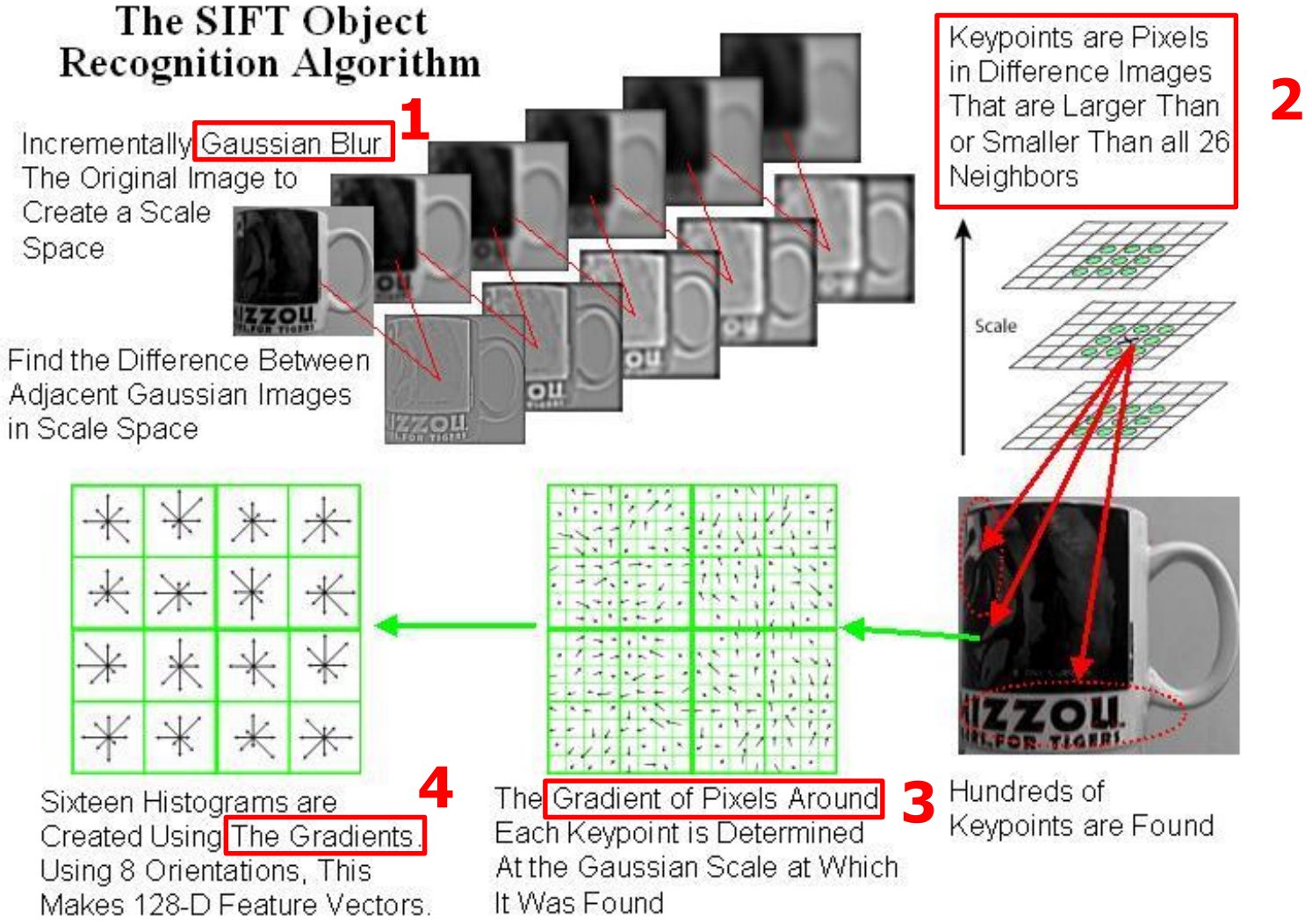
- Diviser l'image d'entrée en plusieurs zones
- Conserver les *n* features plus « forts »
- Évite de concentrer les features dans une seule région
  - réduit l'erreur de localisation

ex. *n=3*

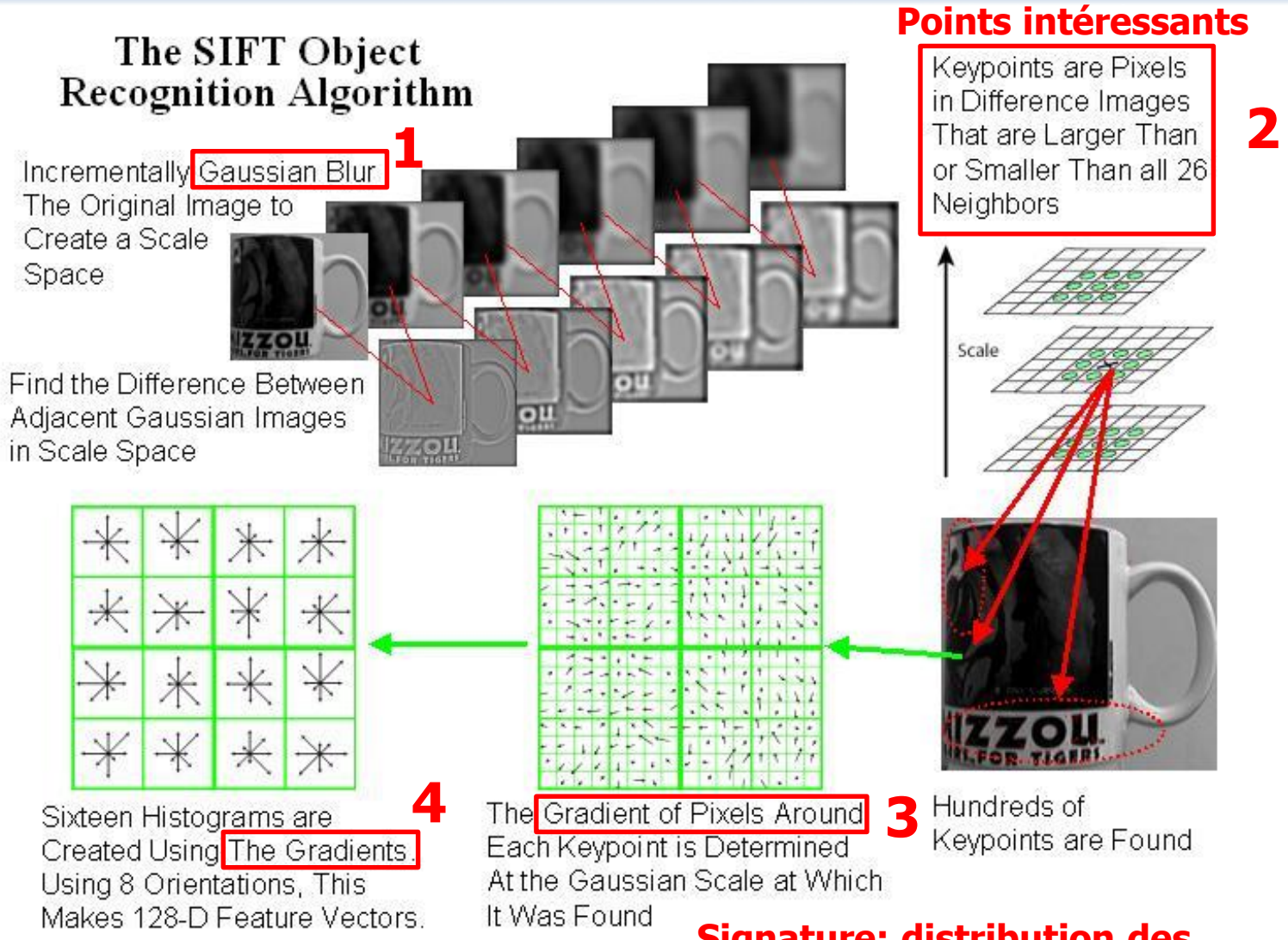


Exemples de descripteurs : SIFT, SURF, BRIEF, ORB

SIFT: Scale Invariant Feature Transform



SIFT: Scale Invariant Feature Transform



Points repères naturels : SIFT

- Correspondance entre points de vue différents

SURF (2008)

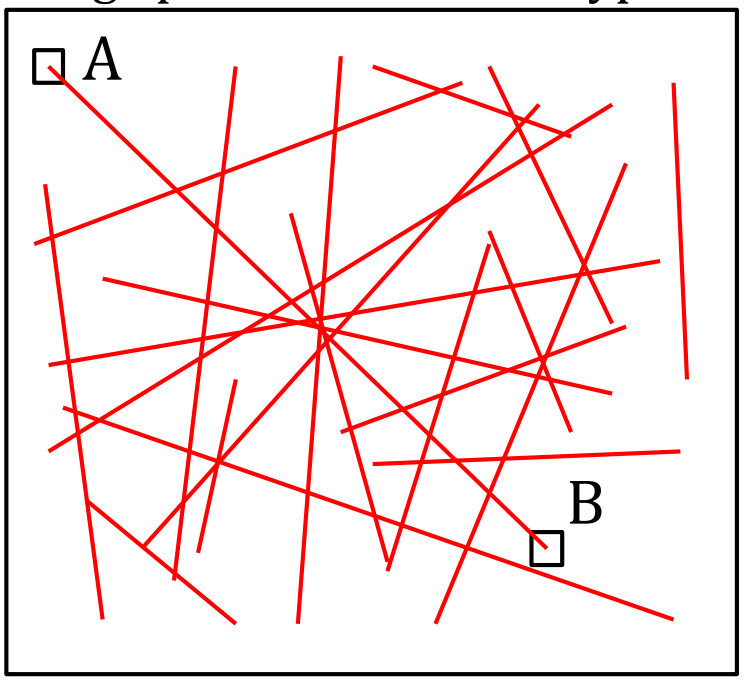
- Speeded up robust features
- En partie inspiré de SIFT
- Plusieurs fois plus rapide

Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, Speeded-Up Robust Features (SURF), Computer Vision and Image Understanding, Volume 110, Issue 3, 2008.

Descripteur BRIEF

- **B**inary **R**obust **I**ndependent **E**lementary **F**eatures
- Choisir un patron fixe de paires de points
- 1 si intensité A > B, 0 sinon
- 128, 256 ou 512 paires, au hasard
- Distance entre deux descripteur : Hamming

Image patch centré sur keypoint




```
D1: 1 0 0 0 1 0 1 1
D2: 1 0 1 1 0 0 1 0
```

4

```
D1⊕D2: 0 0 1 1 1 0 0 1
```

(xor)

```
POPCNT: 4
```

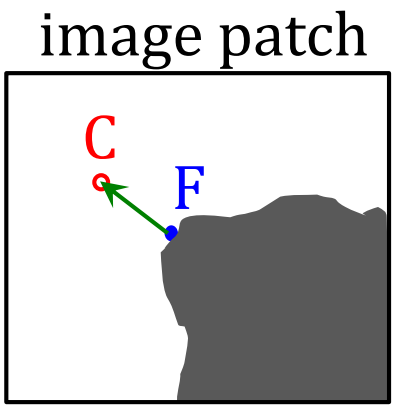
population count

Deux instructions machines SSE (taille registre jusqu'à 512 bits)

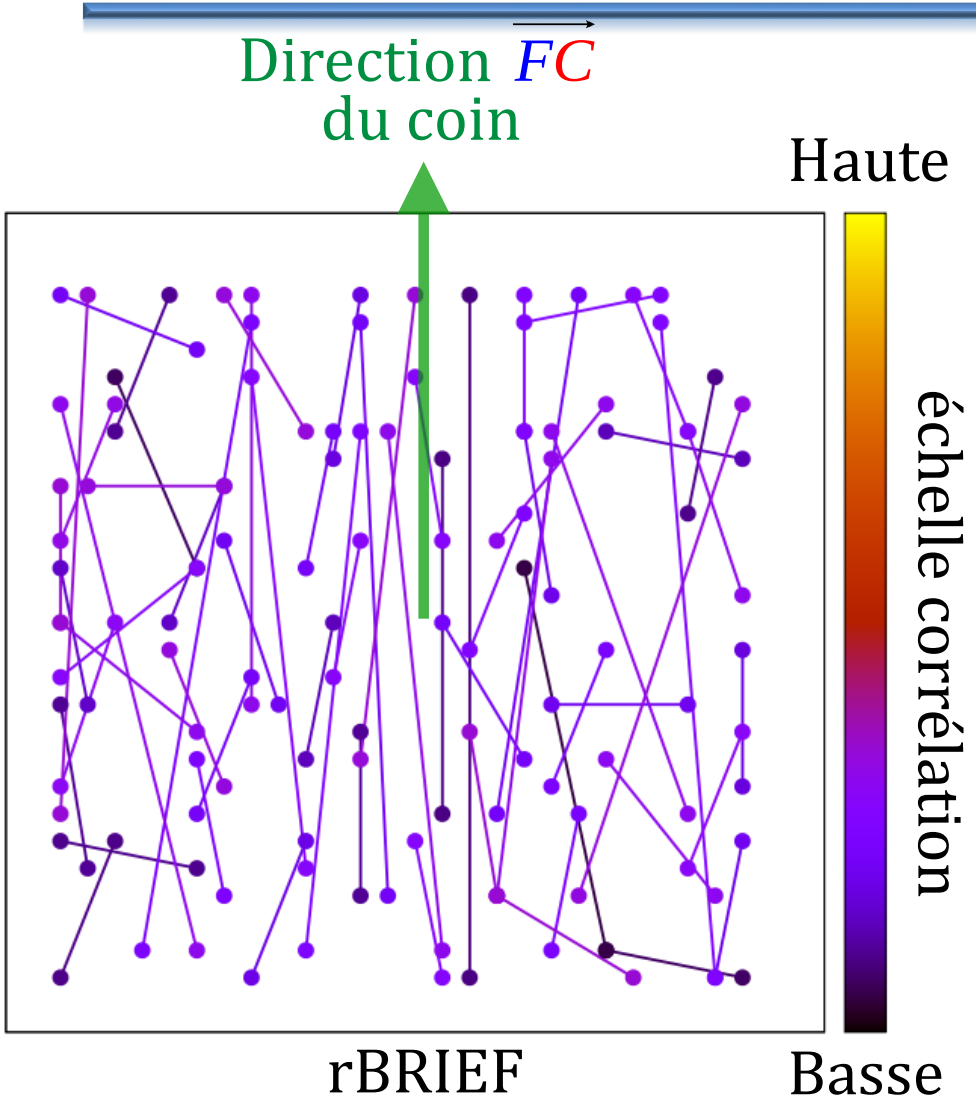
M. Calonder, V. Lepetit, C. Strecha, P. Fua. BRIEF: Binary Robust Independent Elementary Features, *11th European Conference on Computer Vision (ECCV)*, 2010.

ORB feature

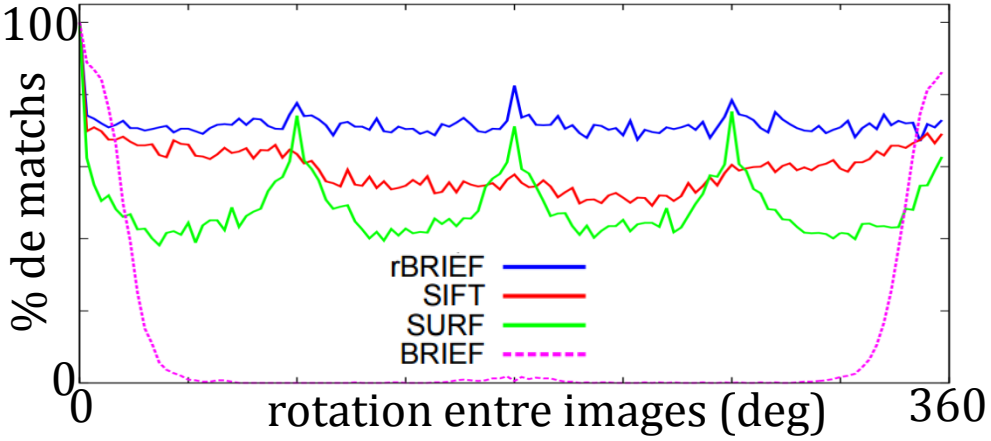
- **O**riented FAST and **R**otated **B**rief
- (1) Besoin de trouver une direction « dominante » d'un coin F : Oriented FAST
  - moment d'une image $m_{pq} = \sum_{x,y} x^p y^q I(x,y)$
  - centre de masse $C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$
  - direction du coin : $\overrightarrow{FC}$



ORB : oFAST + rBRIEF



- Apprentissage d'un *rotated BRIEF* dé-corrélé
- Plus robuste que SIFT aux rotations :



ORB: An efficient alternative to SIFT or SURF, Rublee, E. et al, *ICCV* 2011.

ORB: Comparaison temps exécution

The ORB system breaks down into the following times per typical frame of size 640x480. The code was executed in a single thread running on an Intel i7 2.8 GHz processor:

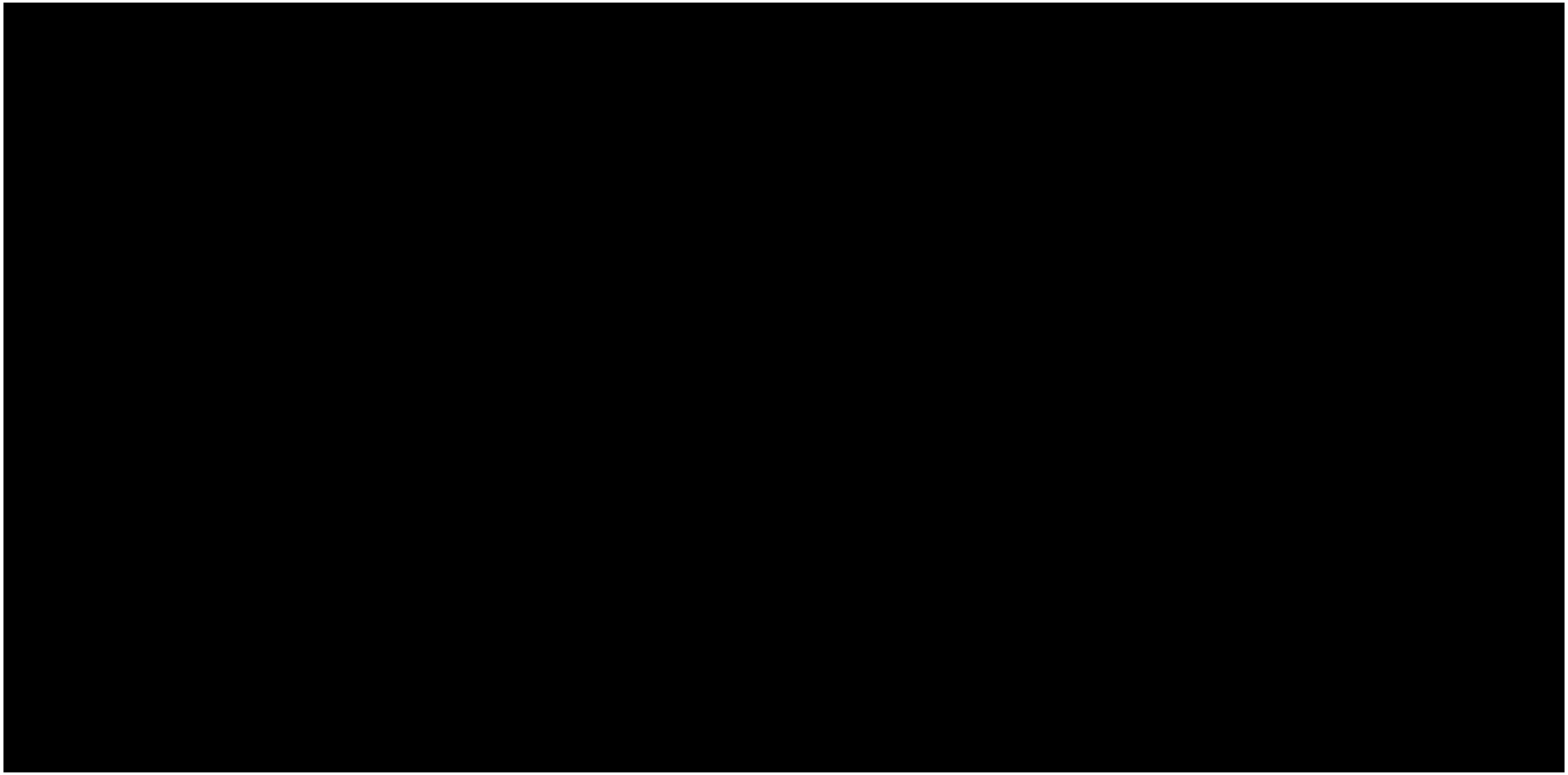
ORB:	Pyramid	oFAST	rBRIEF
Time (ms)	4.43	8.68	2.12

When computing ORB on a set of 2686 images at 5 scales, it was able to detect and compute over 2×10^6 features in 42 seconds. Comparing to SIFT and SURF on the same data, for the same number of features (roughly 1000), and the same number of scales, we get the following times:

Detector	ORB	SURF	SIFT
Time per frame (ms)	**15.3**	217.3	5228.7

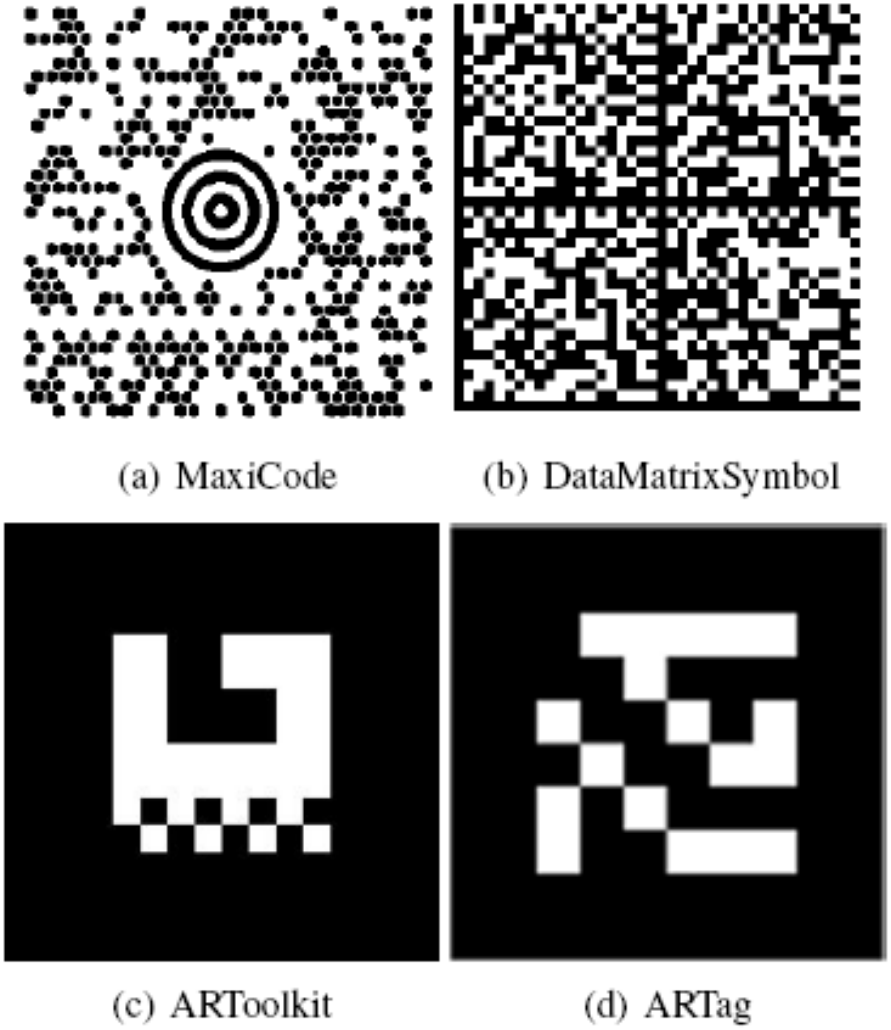
These times were averaged over 24 640x480 images from the Pascal dataset [9]. ORB is an order of magnitude faster than SURF, and over two orders faster than SIFT.

ORB-SLAM2



Points repères artificiels : *Fiducial Marker*

- Repères artificiels
- Grands contrastes :
  - bords
  - coins
- Facile à détecter
- Peut contenir données



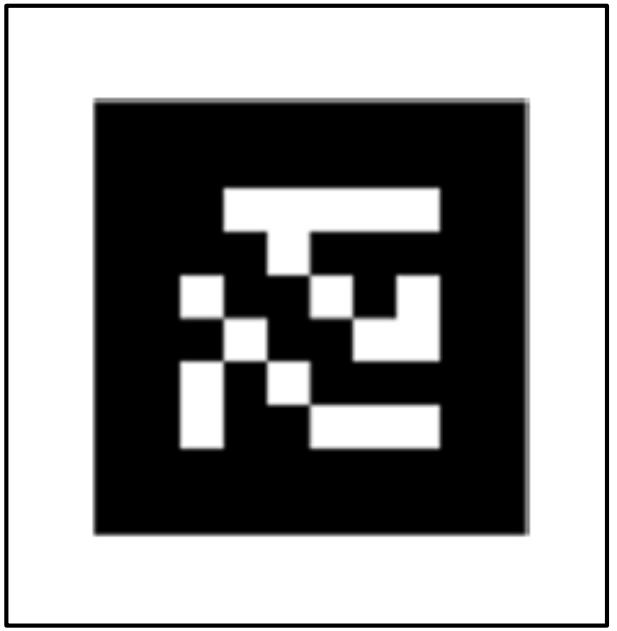
(a) MaxiCode (b) DataMatrixSymbol

(c) ARToolkit (d) ARTag

ArTag: *Fiducial Marker*

- Développé par M. Fiala
- Contient code numérique + redondance
- Retrouve position + orientation

N'oubliez-pas la bordure blanche!



ALVAR (similaire à ArTag)

- Version GNU
- Fonctionne sur Windows ou Linux
- Dépend juste d'OpenCV
- `http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html`