ROBOTIQUE
-ELE4203-

Cours #9: Vision artificielle: techniques et applications (partie 1)

Enseignant: Jean-Philippe Roberge

ÉCOLE POLYTECHNIQUE MONTRÉAL

Cours #9

- **Résultats du sondage informel du dernier cours**
- **Présentation du *robot de la semaine***
- **Bref retour sur l'examen (remise des copies et revue des questions)**
- **Retour sur les notions du cours #8:**
  - Déplacement en mode point par point
    - En définnissant un profil de vitesse trapézoïdale (*LSPB*)
    - Trajectoires *Bang-bang* **
    - À l'aide de polynômes d'ordre 3 **
    - À l'aide de polynômes d'ordre 5 **

Cours #9

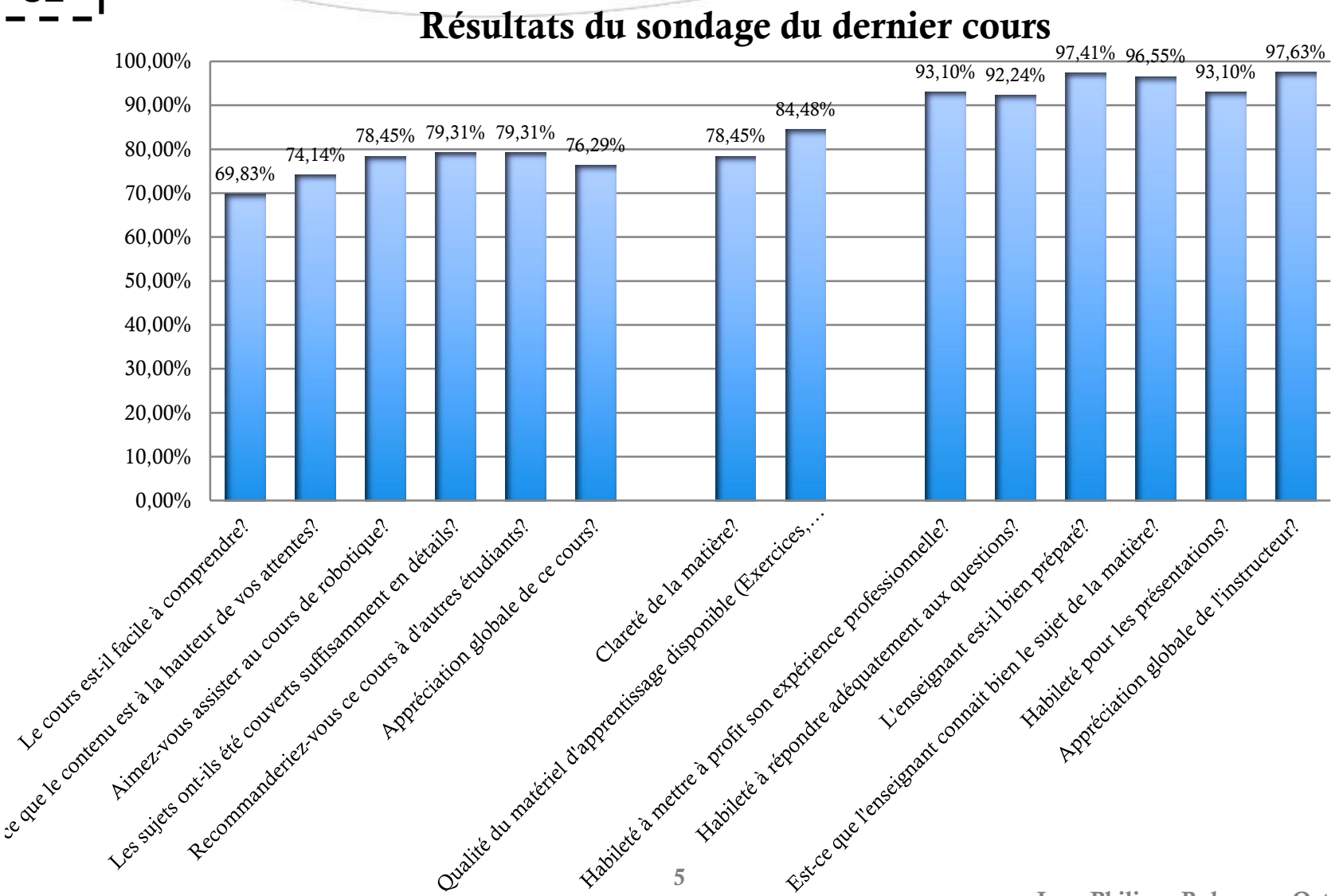
- **Revue des notions du cours #8**
  - Déplacement en mode continu: trajectoire définie par plusieurs points
    - Splines cubiques
- **Début de la matière sur la vision par ordinateur**
  - Pourquoi est-ce que la vision est importante en robotique?
  - Introduction et discussion sur la manière de représenter une image
  - Introduction à différentes techniques de traitement d'images
  - Le rôle de l'éclairage

Cours #9

- **Notions de traitement de bas niveau**
  - Relations entre pixels: voisinage et distance
  - Correction d'histogramme (nb pixels / niveau d'intensité)
    - Étirement, décalage, inversion, seuillage, etc…
    - Seuillage automatique à l'aide d'Otsu
  - Traitement du bruit de type poivre et sel
  - Application de masque
  - Exemple et petite démonstration MATLAB
  - **Présentation d'un intérêt: course automobile**

Résultat du sondage (1)

n=31



Résultats du sondage (2)

- ***Quelques-uns des commentaires pertinents reçus (cours):***
  - *"Plus de robots: genre brève présentation d'un robot funky à chaque début de cours"*
  - *"Consacrer un peu plus de temps aux exercices"*
  - *"Le cours s'oriente plus pour des génies meca. En tant que […] je m'attendais à plus de programmation.*
  - *"Il faudrait faire un exercice de cinématique inverse complet en classe à plus de 3 DDL avant le laboratoire de cinématique inverse"*
  - *Souhaité de mieux pour ce cours: "Plus de temps pour assimiler la cinématique inverse"*
  - *Souhaité de mieux pour ce cours: "Plus d'exercices concernant la cinématique inverse"*
  - *Souhaité de mieux pour ce cours: "Moins de mécanique, plus de contrôle, plus de vision"*

Résultats du sondage (3)

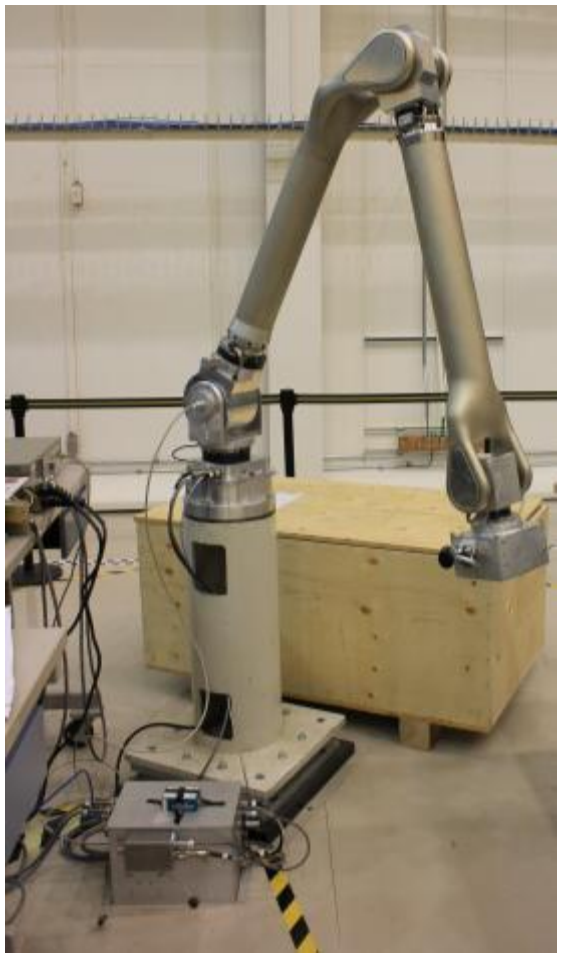
- ***Commentaires labos:***
  - *"Diminuer le nombre de labos et les remplacer avec des labos beaucoup plus pratiques où on a la possibilité de manipuler plus"*
  - *"Les labos sont trop théoriques et calculatoires. Je m'attendais à manipuler les robots réels […] De plus, il y a des erreurs dans les énoncés de lab"*
  - *"On trouve assez souvent des erreurs dans les énoncés de lab"*

Robot de la semaine:

ESI Robotic Arm

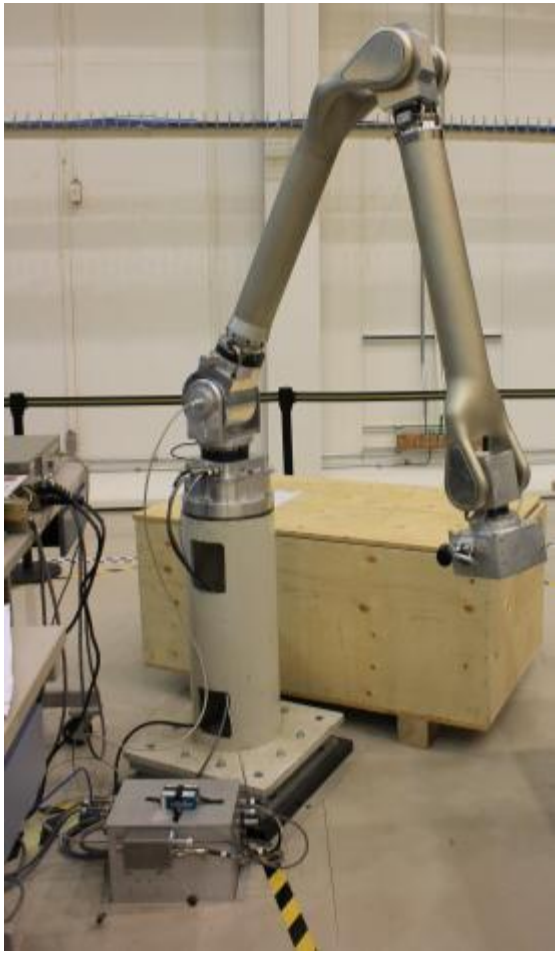
Robot de la semaine (1)

- Voici un robot différent de ce que vous retrouverez généralement en industrie:



Robot de la semaine (2)

- Quels sont les caractéristiques particulières de ce robot?



Robot de la semaine (3)

- Robot présenté au dernier cours dans le cadre du suivi de trajectoire:

- Visitons un exemple de nuage de points 3D.

Retour sur l'examen

Retour sur l'examen (1)

Question 1 (2 points) Soit un ensemble de repères positionnés tel qu'illustré à la figure 1 :

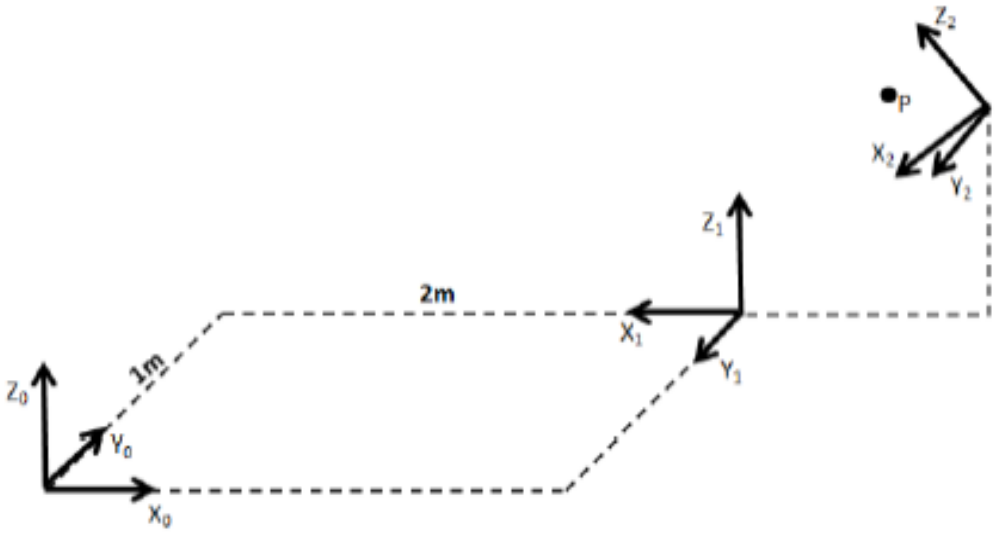


FIGURE 1 – Positionnement des référentiels

Avec le point P exprimé dans le repère 2, tel que : ${}^{2}P = [0.1, 0.1, 0.1]$. Sachant que :

$$ {}^{0}T_{2} = \begin{bmatrix} -0.707 & 0 & -0.707 & 3 \\ 0 & -1 & 0 & 1 \\ -0.707 & 0 & 0.707 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

Donnez les coordonnées du point exprimé dans le repère 1 : ${}^{1}P = [{}^{1}p_x, {}^{1}p_y, {}^{1}p_z,]$

Retour sur l'examen (2)

Question 2 (10 points) Soit un robot RRP dont les paramètres de Denavit-Hartenberg sont les suivant :

i	θ	d	a	α
1	θ_{1v}	d_1	0	90°
2	θ_{2v}	0	a_2	−90°
3	0	d_{3v}	0	0

Les matrices transformation homogènes sont données par :

$$ {}^0T_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

$$ {}^1T_2 = \begin{bmatrix} C_2 & 0 & -S_2 & a_2C_2 \\ S_2 & 0 & C_2 & a_2S_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3) $$

$$ {}^2T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{3v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4) $$

$$ {}^1T_2^2T_3 = \begin{bmatrix} C_2 & 0 & -S_2 & a_2C_2 - d_{3v}S_2 \\ S_2 & 0 & C_2 & a_2S_2 + d_{3v}C_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5) $$

$$ {}^0T_1^1T_2^2T_3 = \begin{bmatrix} C_1C_2 & -S_1 & -C_1S_2 & C_1(a_2C_2 - d_{3v}S_2) \\ S_1C_2 & C_1 & -S_1S_2 & S_1(a_2C_2 - d_{3v}S_2) \\ S_2 & 0 & C_2 & d_1 + d_{3v}C_2 + a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6) $$

a) Faites un schéma de ce robot (1 pt)

b) Calculez la matrice Jacobienne de ce robot (4 pts)

c) Développer le modèle cinématique inverse (5 pt)

Retour sur l'examen (3)

Question 3 (3 points)

Vous travaillez pour une compagnie qui manufacture un robot RPR dont les variables articulaires sont θ_1, L_3v et θ_3. La cinématique directe de ce robot est donnée par :

$$ {}^0T_3 = \begin{bmatrix} C_{13} & -S_{13} & 0 & C_{13}L_4 + C_1L_2 \\ S_{13} & C_{13} & 0 & S_{13}L_4 + S_1L_2 \\ 0 & 0 & 1 & L_{3v} + L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

Vous êtes en charge de développer un logiciel qui permettra aux clients de spécifier la pose désirée du robot dans l'espace de travail, par rapport au repère de base. En discutant avec vos collègues, certains font des suggestions sur la manière d'aborder le problème de conception de votre logiciel :

- Le collègue A vous suggère de permettre au client de spécifier la pose en utilisant les coordonnées x, y, z.
- Le collègue B vous suggère de permettre au client de spécifier la pose en utilisant les coordonnées x, y ainsi que l'orientation de l'effecteur θ_z.
- Le collègue C vous suggère de permettre au client de spécifier la pose en utilisant les coordonnées x, y, z ainsi que l'orientation de l'effecteur θ_z.
- Le collègue D vous suggère de permettre au client de spécifier la pose en utilisant la coordonnée z ainsi que l'orientation de l'effecteur θ_z.

Pour chacune des quatre suggestions ci-dessus, dites si la suggestion est valide ou non, et expliquez pourquoi. (4 X 0.75 pts)

Retour sur l'examen (4)

Question 4 (1.5 point)

Vous planifiez la trajectoire de l'effecteur d'un robot planaire RR dont vous connaissez la jacobienne inversée :

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{8}$$

$$J^{-1} = \frac{1}{a_1 a_2 S_2} \begin{bmatrix} a_2 C_{12} & a_2 S_{12} \\ -a_1 C_1 - a_2 C_{12} & -a_1 S_1 - a_2 S_{12} \end{bmatrix} \tag{9}$$

D'après vous, y a-t-il des configurations articulaires $[\theta_1, \theta_2]$ à éviter lorsque vous planifiez une trajectoire pour un tel robot ?

Retour sur l'examen (5)

Question 5 (3.5 points)

Soit le robot RPR de la figure suivante :

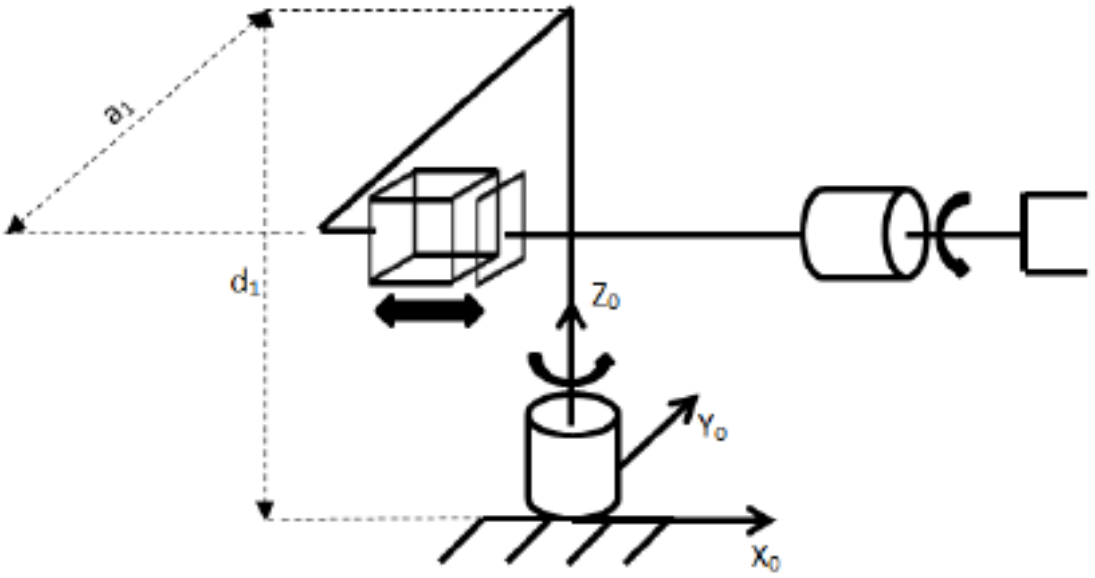


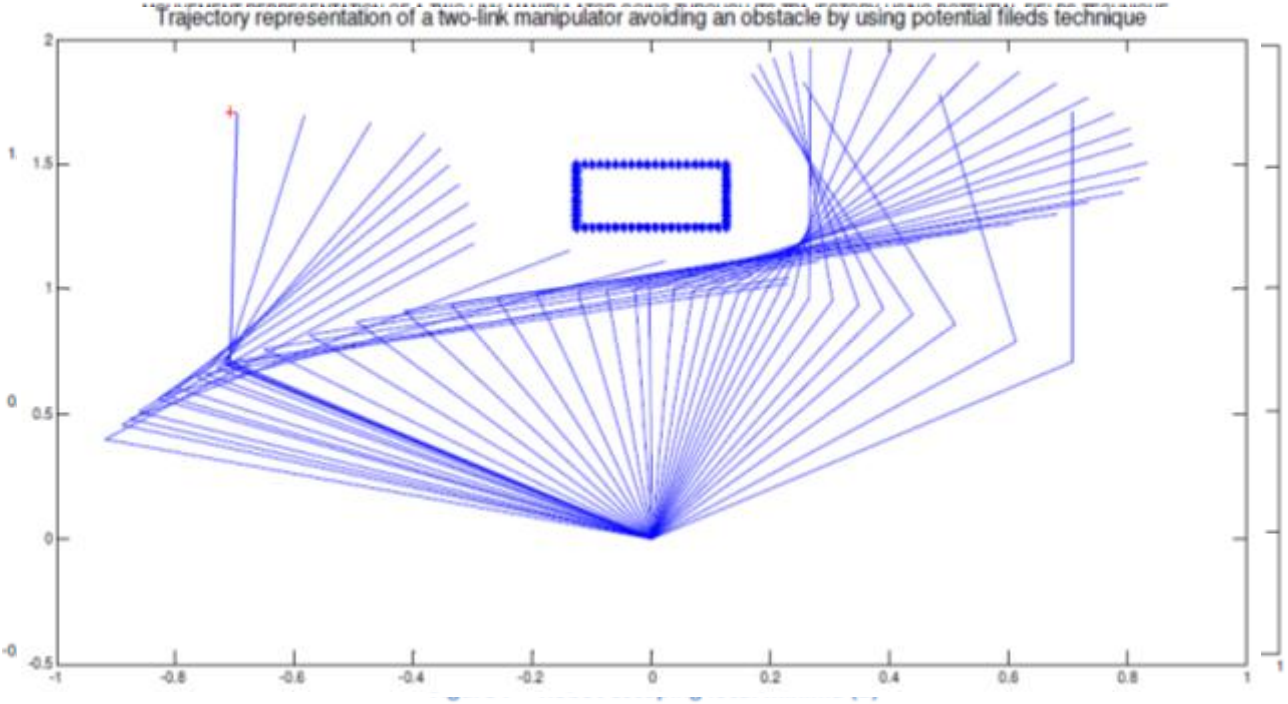
FIGURE 2 – Robot RPR

- Donner le tableau des paramètres de Denavit-Hartenberg pour ce robot (spécifier et définir, si nécessaire, les paramètres manquants sur le schéma).
- Donner ${}^0T_1, {}^1T_2, {}^2T_3$

Retour sur le cours #8
Planification de trajectoires (1)

- Voici une notion qui **n'est pas à l'ordre du cours**, il s'agit de l'utilisation de champs de potentiels pour diriger les robots.

- Lorsqu'il y a des obstacles dans l'environnement du robot, c'est une approche relativement simple conceptuellement qui consiste à utiliser un champ potentiel de répulsion et un champ d'attraction:



Cours #8: Planification de trajectoires (2)
Déplacement en mode point-par-point - LSPB

- Pour ce mode, voici les profils typiques de position, vitesse et accélération entre deux points q_o et q_1:

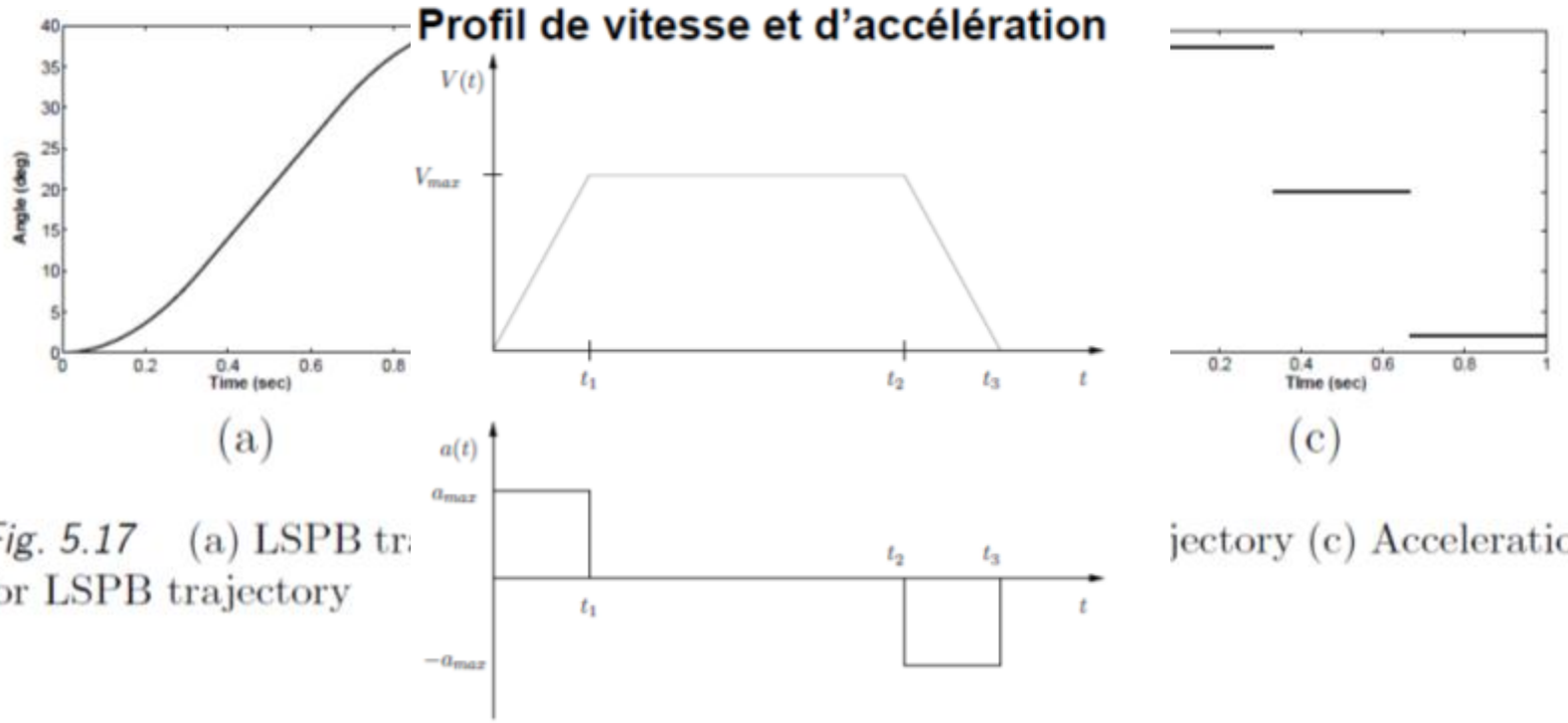


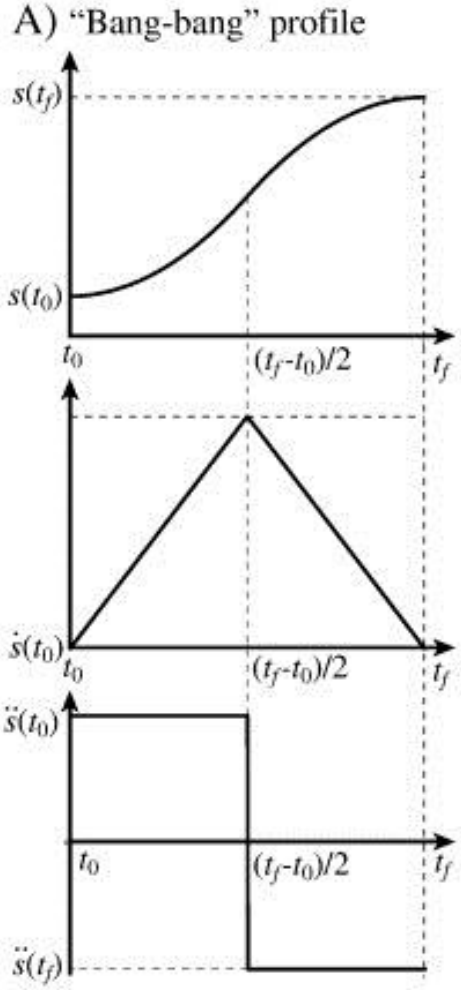
Fig. 5.17 (a) LSPB tr… jectory (c) Acceleration for LSPB trajectory

Cours #8: Planification de trajectoires (3)
Déplacement en mode point-par-point – Trajectoire Bang-bang

- Dans un contexte où:
  - Les mouvements saccadés ne sont pas un problème pour le système robotique et les produits manipulés
  - La rapidité est très importante
- On peut considérer, au lieu du LSPB, d'élaborer des trajectoires à temps minimal.
  - Les trajectoires à temps minimal sont les trajectoires les plus rapides pour amener le robot d'une configuration articulaire q_0 à une configuration articulaire q_1.

Cours #8: Planification de trajectoires (4)
Déplacement en mode point-par-point – Trajectoire Bang-bang

- De plus, le profil de vitesse n'est pas nécessairement trapézoïdal:



Il sera trapézoïdale seulement si V_{max} est atteinte lors du déplacement.

Cours #8: Planification de trajectoires (5)
Déplacement en mode point-par-point – Polynômes cubiques

- Une autre approche pour définir des trajectoires en mode point-par-point consiste à utiliser des polynômes d'ordre 3:

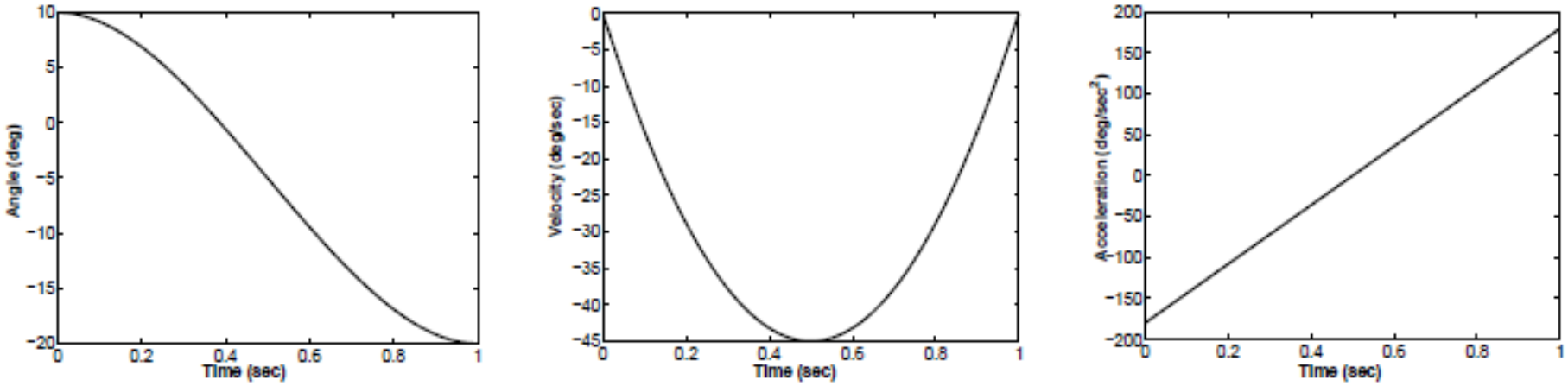
$$q_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- Ces polynômes ont quatres paramètres (quatre degrés de liberté) et on peut donc, en général, leur imposer quatre contraintes.
  - On peut par exemple, donner des contraintes sur la position initiale et finale (2) ainsi que sur la vitesse initiale et finale (2)

- La vitesse articulaire est donnée simplement en dérivant:

$$\dot{q}_d(t) = a_1 + 2a_2 t + 3a_3 t^2$$

Cours #8: Planification de trajectoires (6)
Déplacement en mode point-par-point – Polynômes cubiques



(a) (b) (c)

Fig. 5.13 (a) Cubic polynomial trajectory (b) Velocity profile for cubic polynomial trajectory (c) Acceleration profile for cubic polynomial trajectory

Cours #8: Planification de trajectoires (7)
Déplacement en mode point-par-point – Polynômes quintiques

- Les polynômes quintiques sont des polynômes de 5ième ordre de la forme:

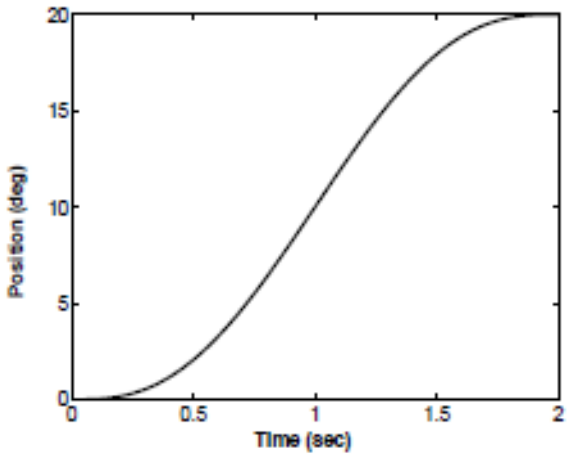
$$q_d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

- Puisque le polynôme possède 6 paramètres, il est possible, en général, de lui donner 6 contraires:
  - 2 pour la position initiale et finale
  - 2 pour la vitesse initiale et finale
  - 2 pour l'accélération initiale et finale

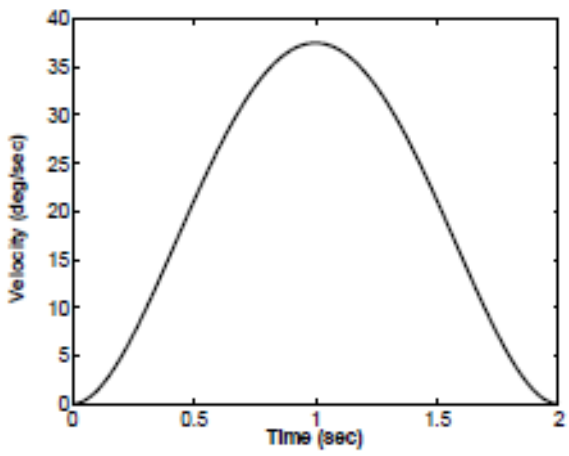
- La vitesse et l'accélération articulaires sont données par:

$$\dot{q}_d(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$
$$\ddot{q}_d(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

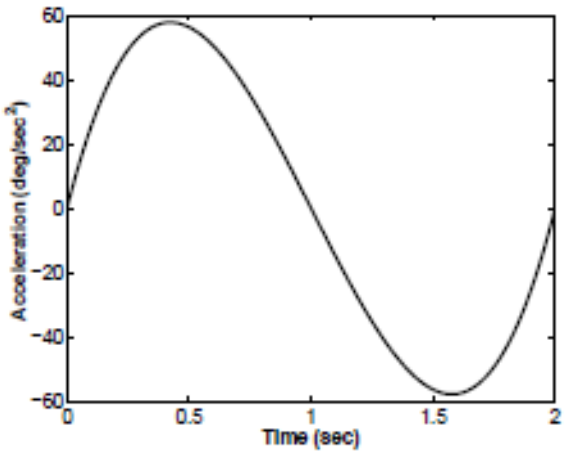
Cours #8: Planification de trajectoires (8)
Déplacement en mode point-par-point – Polynômes quintiques



(a)



(b)



(c)

Fig. 5.15 (a) Quintic Polynomial Trajectory. (b) Velocity Profile for Quintic Polynomial Trajectory. (c) Acceleration Profile for Quintic Polynomial Trajectory

Cours #8: Planification de trajectoires (9)
Déplacement en mode continu – Spline cubique

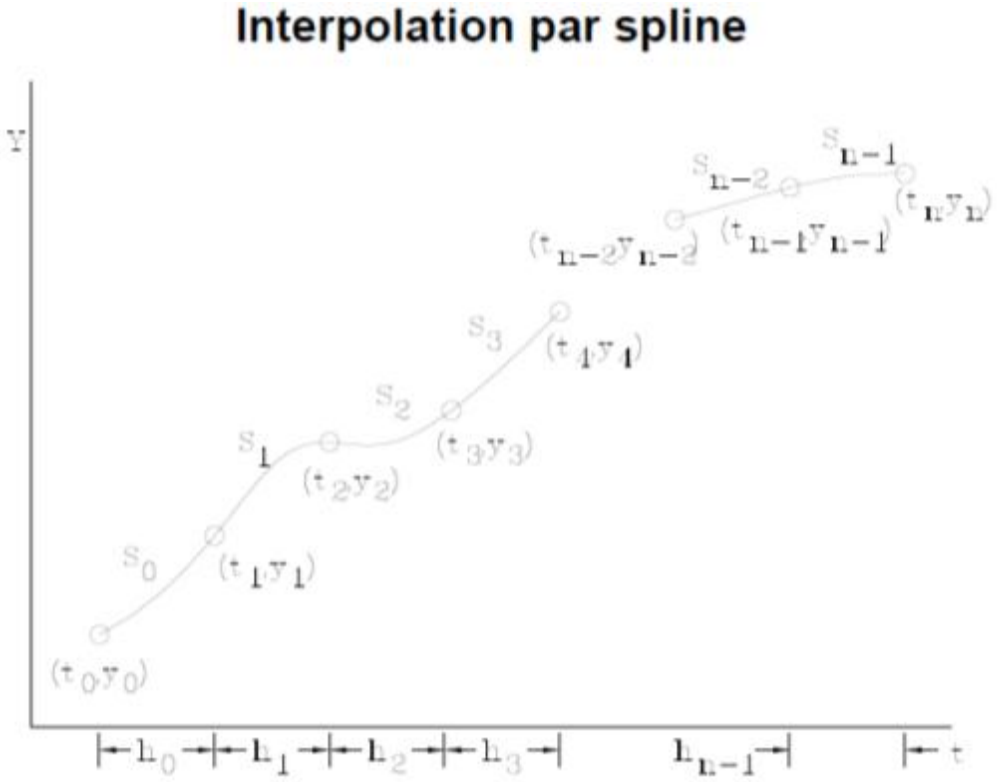
- Lorsque le robot fonctionne en mode continu, les points à atteindre ne sont plus des points singuliers situés à une distance considérable.
  - Il s'agit maintenant de points très rapprochés et nous devons donc considérer des trajectoires définies différemment.
    - En particulier, puisque plusieurs points seront interpolés en peu de temps, il serait pratique d'avoir des trajectoires connectant ces points qui seraient continues en vitesse et en accélération aux points de jonction.
    - Pour ce faire, nous utiliserons une ***spline cubique***.

- Une spline, c'est une fonction définie par morceaux à l'aide de polynômes.
  - Une spline cubique est donc une fonction définie à l'aide de polynômes d'ordre 3 de forme:

$$q_i(t) = a_i + b_i(t - t_0) + c_i(t - t_0)^2 + d_i(t - t_0)^3$$

Cours #8: Planification de trajectoires (10)
Déplacement en mode continu – Spline cubique

- Donc, pour interpoler *n+1* points nous utiliserons *n* polynômes qui seront continus en vitesse et en accélération aux points de jonctions.



- Comment peut-on construire la spline cubique avec ces conditions de continuité?

Cours #8: Planification de trajectoires (11)
Déplacement en mode continu – Spline cubique

- Énonçons de façon plus clair les contraintes définissant la spline cubique:
  - Soit une fonction f(t) définie dans l'intervalle [a,b] avec $a=t_0 < t_1 < \ldots < t_n = b$ et un ensemble de points prédéterminés, alors:

$1-S$ est la spline cubique, donné par S_j, dans l'intervalle $[t_j, t_{j+1}]$ pour chaque $j = 0,1,\ldots,n-1$

$2-S_j(t_j) = f(t_j) = y_j$ pour chaque $j = 0,1,\ldots,n$

$3-S_{j+1}(t_{j+1}) = S_j(t_{j+1})$ pour chaque $j = 0,1,\ldots,n-2$

$4-\frac{d}{dt}S_{j+1}(t_{j+1}) = \frac{d}{dt}S_j(t_{j+1})$ pour chaque $j = 0,1,\ldots,n-2$

$5-\frac{d^2}{dt^2}S_{j+1}(t_{j+1}) = \frac{d^2}{dt^2}S_j(t_{j+1})$ pour chaque $j = 0,1,\ldots,n-2$

6 – On impose les vitesse aux extrémités de la spline:

$$\frac{d}{dt}S_0(t_0) = \frac{d}{dt}f(t_0) \quad \text{et} \quad \frac{d}{dt}S_n(t_n) = \frac{d}{dt}f(t_n)$$

Cours #8: Planification de trajectoires (12)
Déplacement en mode continu – Spline cubique

- Ces équations peuvent être résolues sous forme matricielle:

$$Ax = b$$

$$\boldsymbol{A} = \begin{bmatrix} 2h_0 & h_0 & \cdots & \cdots & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} \frac{3}{h_0}(a_1 - a_0) - 3\frac{d}{dt}f(t_0) \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 3\frac{d}{dt}f(t_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1}) \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} c_0 & \cdots & c_n \end{bmatrix}^T$$

Cours #8: Planification de trajectoires (13)
Déplacement en mode continu – Spline cubique

- Méthode de résolution:
  - Vous connaissez les a_j quasi-directement, vous connaissez les h_j et les pentes imposées au début et à la fin des splines, donc:
    - Vous notez les a_j
    - Vous posez le système d'équation sous forme matricielle et résolvez pour trouver les c_j
    - Vous trouvez les b_j et les d_j à l'aide des relations établies plus tôt, i.e.:

$$b_j = \frac{1}{h_j}\left(a_{j+1} - a_j\right) - \frac{h_j}{3}\left(2c_j + c_{j+1}\right)$$

$$d_j = \frac{c_{j+1} - c_j}{3h_j}$$

Cours #8: Planification de trajectoires (14)
Robots mobiles – Un aperçu

- En effet, étant donné que le robot se déplace et que l'environnement n'est pas connu *a priori*, il faut user de techniques fort différentes.
  - Entres autres, il faut utiliser des capteurs pour saisir les formes de l'environnement à traver lequel le robot se déplacera.
  - Il faut tenter de localiser le robot dans un repère de référence.
  - Il faut éviter les obstacles qui ne sont pas non plus connus *a priori*
  - La minimisation de l'énergie revêt ici une importance grandiose(!) en général
  - Il faut enfin créer une trajectoire navigable pour le robot mobile.
    - Ensuite un contrôleur se chargera de commander le robot afin qu'il assure un bon suivi de trajectoire.

Cours #9
- Vision -

Cours #9
Introduction à la vision (1)

- **En général, pourquoi ayons-nous recours à la vision artificielle?**
  - En général on peut noter deux grandes catégories d'applications:
    - L'amélioration de l'information pictorale pour l'interprétation des images par les humains.
    - Le traitement des données d'image pour le stockage, la transmission et/ou la représentation de l'environnement pour des systèmes autonomes.

"One picture is worth more than ten thousand words".

- **Pourquoi enseigner la vision artificielle dans un cours de robotique?**
  - Lorsqu'un robot interagit avec l'environnement, ce dernier doit être capable de percevoir son environnement.
  - La vision artificielle au sens large constitue l'un des moyens de perception des plus puissants qu'il existe actuellement.
- **Dans ce cours, nous présenterons une introduction à quelques techniques inhérentes au traitement d'images.**

Cours #9
Introduction à la vision (2)

- **Tout d'abord, qu'est-ce que le traitement d'image?**

- **Tout d'abord, qu'est-ce que le traitement d'image?**
  - Une image (représentée en tons de gris) peut êre définie comme une fonction multivariable :

$$I(x, y)$$

  - Où x et y sont simplement les coordonnées (planaires) de chaque éléments qui constituent l'image, et l'amplitude de *I* pour chaque paire (x,y) est l'*intensité* ou le *niveau de gris* de l'image à ce point.

Cours #9
Introduction à la vision (3)

- **Tout d'abord, qu'est-ce que le traitement d'image (suite)?**

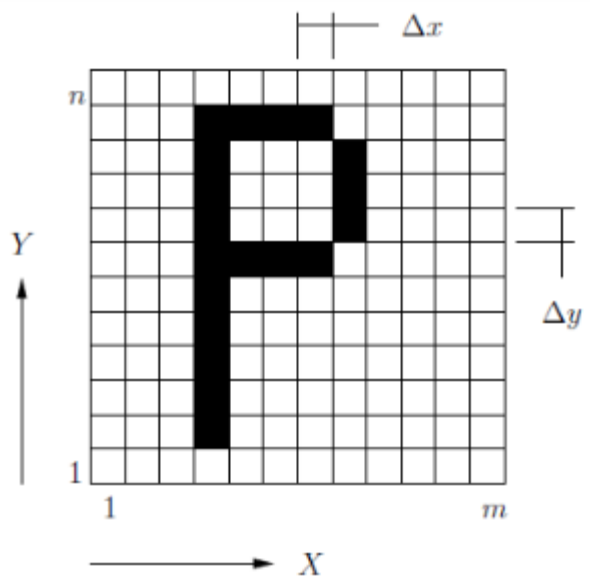
$$I(x, y)$$

- **Lorsque *I*, *x*, *y* ont tous des valeurs finies, on appelle l'image "*image numérique*".**
  - Donc, le traitement d'image réfère ici au traitement d'image numérique à l'aide d'ordinateur et d'algorithmes informatiques.
    - Ces algorithmes informatiques viseront à analyser, voir changer les valeurs de *I* pour certains éléments (*x,y*).
- **Notez qu'une image numérique est constituté d'un certains nombre fini d'éléments, chacun d'eux possédant un lieu (x,y) et une amplitude *I*.**
  - Ces éléments se nomment "éléments d'image", ou en anglais "*picture element*" ou sous forme plus concise: ***pixel*** (pixel pour *pi(x)ture element*)

Culture générale: La généralisation du pixel dans une image 3D se nomme le **voxel** pour "*Volumetric Picture Element*".

Cours #9
Introduction à la vision (4)

- **Autres notions:**
  - Étant donnée que (x,y) constitue le lieu dans le plan de chaque pixel, l'image numérique est de dimension *m* X *n*.
  - Le nombre de pixel en x et en y de l'image numérique se nomme la **résolution**.
    - Ainsi, une image de résolution *m* par *n* possède *m* pixels en x par *n* pixels en y.



Chaque pixel représente la valeur moyenne d'intensité (ou de niveau de gris) d'une surface de dimension Δx X Δy de l'image réelle.

Cours #9
Introduction à la vision (5)

- **Autres notions:**
  - Effet de la résolution:


résolution originale

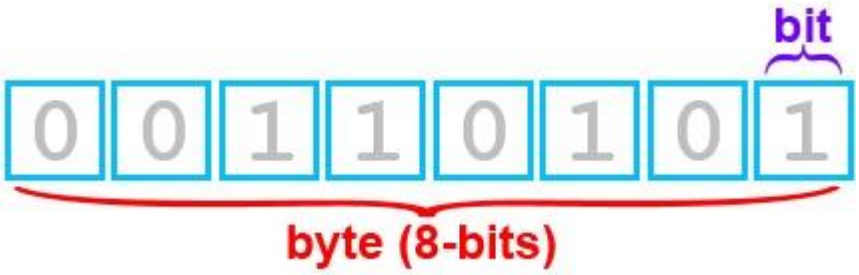

pixel 2× original


pixel 4× original


pixel 8× original

Cours #9
Introduction à la vision (6)

- **Autres notions (suite):**
  - L'intensité de l'image *réelle* sera donc discrétisée en surface, étant donnée que celle-ci sera constituée de pixels.
  - L'intensité sera aussi discrétisée par sa représentation en bits (nombre binaire).



Decimal pattern (Hex Value)	Binary numbers
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10 – (A)	1010
11 – (B)	1011
12 – (C)	1100
13 – (D)	1101
14 – (E)	1110
15 – (F)	1111
16 – (10)	10000

Souvent, un considérera une image numérique en tons de gris représentés en 1-byte (8 bits): 2^8 = 256 niveaux de représentation de gris possibles (0 à 255).

Cours #9
Introduction à la vision (7)

- **Autres notions (suite):**
  - Une image noir et blanc: **1 bit** (0=noir et 1=blanc)
  - Une image monochrome (tons de gris): **b bits** (0=noir et 2^b-1=blanc)
  - Une image en couleur est représentée un peu sous le même principe, c'est-à-dire par un ensemble de bits qui représentent le niveau RGB (rouge, vert et bleu)
  - Évidemment, plus l'image sera représentée par un grand nombre de pixels, et par un nombre de bits élevé, plus l'image numérique représentera fidèlement l'image réelle, en général.

Cours #9
Introduction à la vision (8)

- **Autres notions (suite):**
  - Effet du nombre de bits:

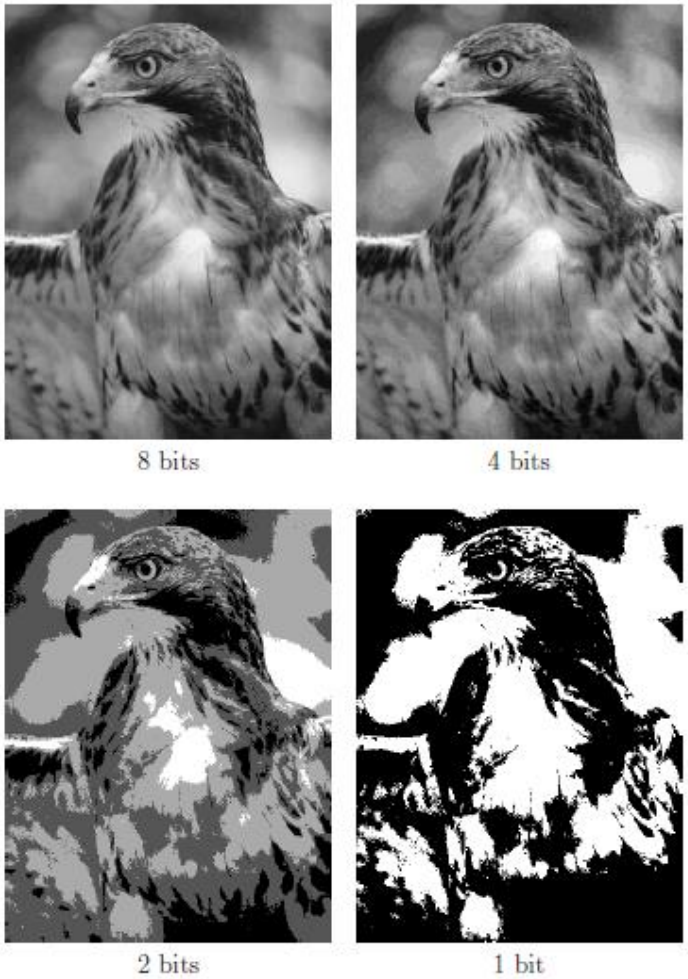


FIG. 1.3 – Influence du nombre de bits pour les niveaux de gris

Cours #9
Introduction à la vision (9)

- **Traitement de l'image numérique:**
  - Deux catégories de traitement de l'image numérique:
    - Le *traitement de bas niveau*: filtrage, réduction de bruit, amélioration du contraste. On transforme l'image brute, on la conditionne pour accomplir le traitement de haut niveau avec succès.
    - Le *traitement de haut niveau*: segmentation, identification, reconnaissance de formes, etc…

Introduction à la vision (10)
Éclairage

- **Éclairage:**
  - Dépendamment des opérations que vous désirez effectuer, vous devrez choisir un **éclairage** approprié. Ici, on présente quatre techniques d'éclairage:
    - L'éclairage **diffus** utilisé pour les objets lisses ayant des surfaces régulières
    - L'éclairage **direct** utilisé souvent pour détecter des défauts de surface.

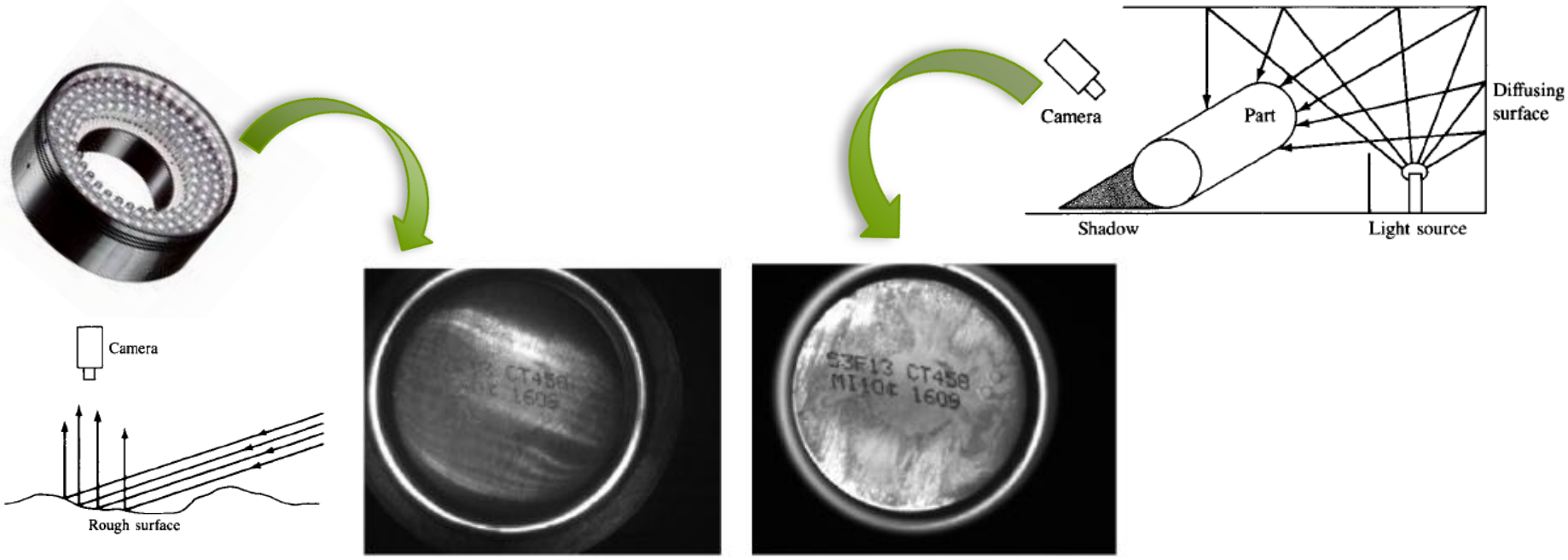
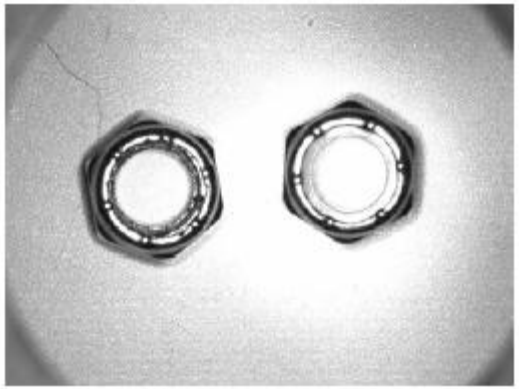
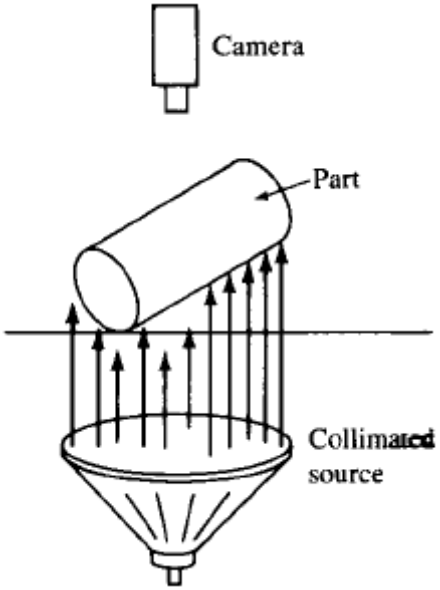


Fig. 6 – Bottom of a soda can. Left: illuminated with a bright field ring light, but shows poor contrast, uneven lighting, and specular reflections. Right: imaged with diffuse light, creating an even background allowing the code to be read.

Introduction à la vision (11)
Éclairage

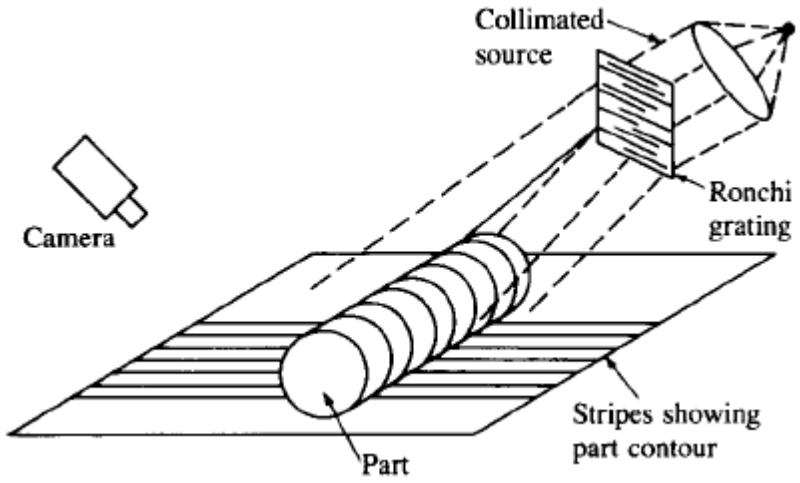
- **Éclairage (suite):**
  - Dépendamment des opérations que vous désirez effectuer, vous devrez choisir un **éclairage** approprié. Ici, on présente quatre techniques d'éclairage:
    - L'éclairage **arrière** qui permet de bien générer des images en noir et blanc





Introduction à la vision (12)
Éclairage

- **Éclairage (suite):**
  - Dépendamment des opérations que vous désirez effectuer, vous devrez choisir un **éclairage** approprié. Ici, on présente quatre techniques d'éclairage:
    - L'éclairage **structuré** par bande ou par grille. La déformation du patron permet d'identifier les caractéristiques de l'objet.



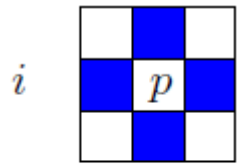
Traitement de bas niveau
Relations entre les pixels (1)

- **Avant d'entrer dans le coeur du traitement de bas niveau, commençons tout d'abord par étudier deux des caractéristiques que possèdent les pixels entre eux: le *voisinnage* et la *distance*.**

- **Voisinnage:**

  - Un pixel p possède 4 voisins horizontaux et verticaux, appellés les 4-voisins de p et notés N_4:

$$(i+1,j) \quad (i-1,j) \quad (i,j+1) \quad (i,j-1)$$

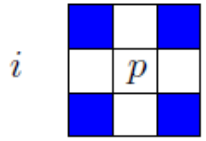


$N_4(p)$

Traitement de bas niveau
Relations entre les pixels (2)

- **Voisinnage:**
  - Le pixel p possède aussi 4 voisins diagonaux, notés N_D:

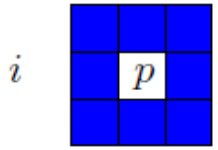
$$(i+1, j+1) \quad (i+1, j-1) \quad (i-1, j+1) \quad (i-1, j-1)$$



$N_D(p)$

  - On peut aussi parler des 8 voisins de p, constitué des 4-voisins de p en plus des voisins diagonaux de p:

$$N_8(p) = N_4(p) \cup N_D(p)$$



$N_8(p)$

Traitement de bas niveau
Relations entre les pixels (3)

- **Distance entre les pixels:**

- Considérons les pixels p p, q et z de coordonnées (i, j), (s, t) et (u, v) respectivement, nous appelons *D* une fonction de distance si:

1. $D(p,q) \geq 0$ avec $D(p,q) = 0$ ssi $p = q$
2. $D(p,q) = D(q,p)$
3. $D(p,z) \leq D(p,q) + D(q,z)$

- **Distance euclidienne:**

$$D_e(p,q) = \sqrt{(i-s)^2 + (j-t)^2}$$

Dans le domaine des $D_e \leq 3$:

			3			
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
3	2	1	0	1	2	3
	$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$	
	$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$	
			3			

Traitement de bas niveau
Relations entre les pixels (4)

- **Distance entre les pixels (suite):**

- **La distance D_4 ou *city-blocks* est donnée par:**

$$D_4(p,q) = |i-s| + |j-t|$$

Dans le domaine des $D_4 \leq 2$:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

- **La distance D_8 ou *chess-board* est donnée par:**

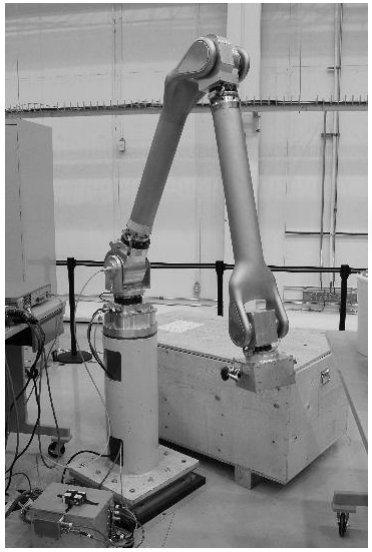
$$D_8(p,q) = \max(|i-s|, |j-t|)$$

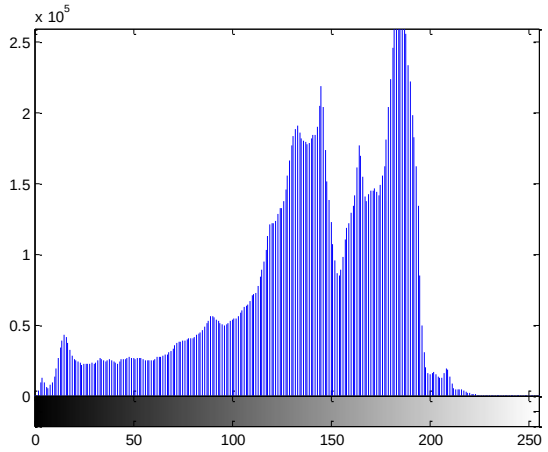
Dans le domaine des $D_8 \leq 2$:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

Traitement de bas niveau
Correction d'histogramme (1)

- La **correction d'histogramme** est la première technique de traitement d'image numérique que nous étudierons.
- Elle permet le *rehaussement* des images.
  - Le *rehaussement* signifie l'amélioration de l'image afin d'en faciliter l'interprétation visuelle et sa compréhension.
  - Dans le contexte de la robotique il signifie, au sens large, l'amélioration des caractéristiques pertinentes pour le traitement de haut niveau propre à l'application dont il est question.





Traitement de bas niveau
Correction d'histogramme (2)

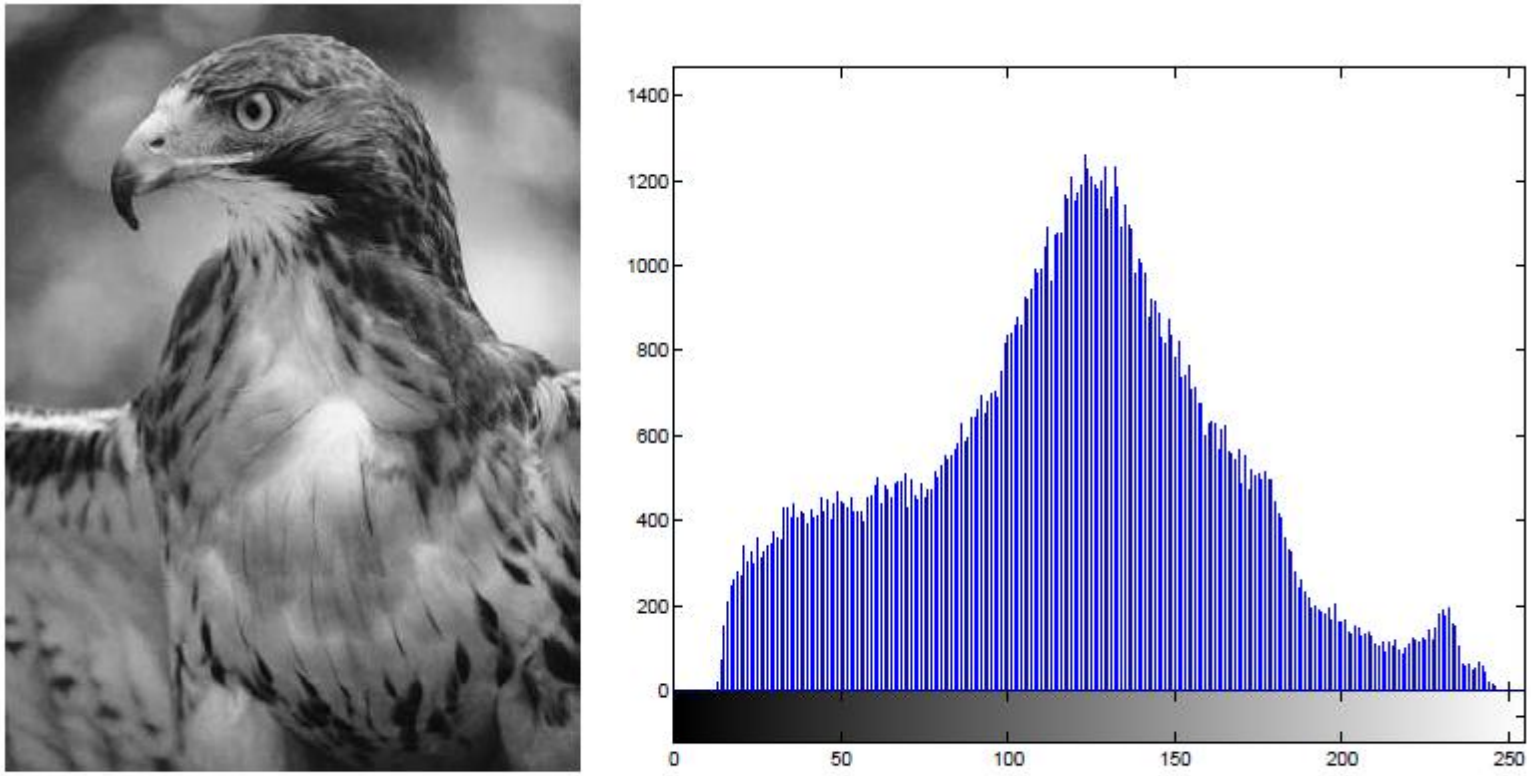
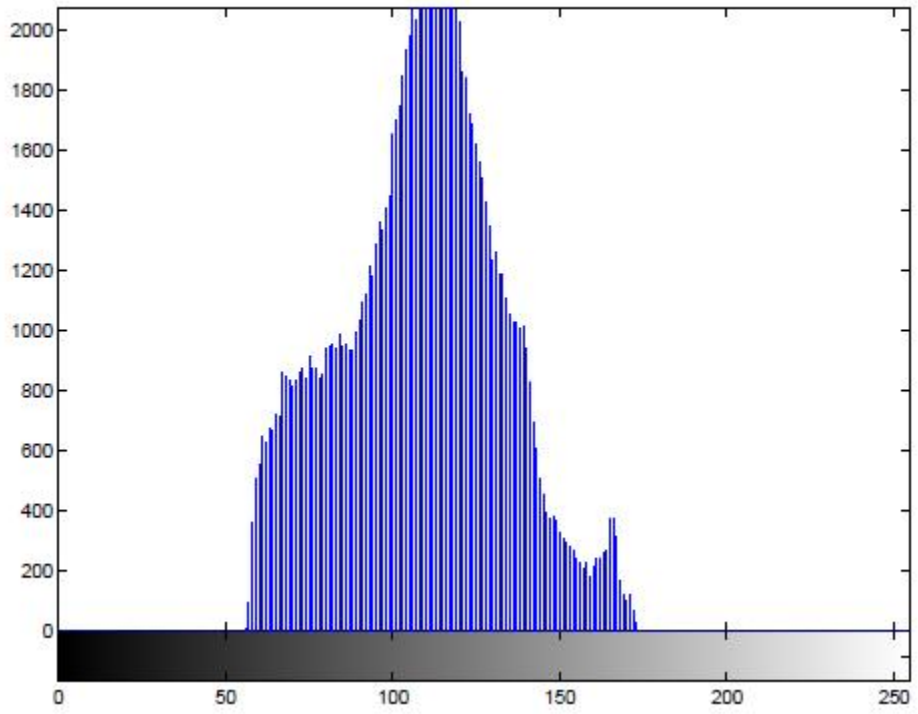


FIG. 2.2 – Histogramme d'une image à 256 niveaux (0 à 255) de gris

Traitement de bas niveau
Correction d'histogramme (3)

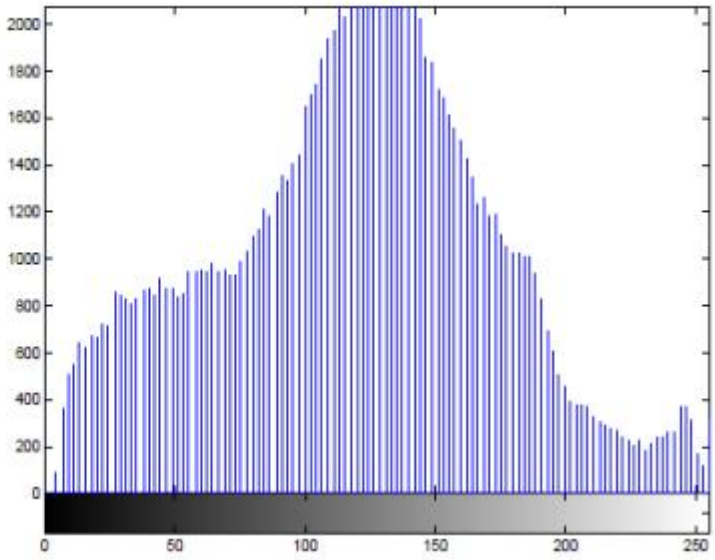
- Considérez la paire image-histogramme suivante:



- Il y a peu de nuances au niveau des tons de gris et cela se voit autant au niveau de l'histogramme qu'au niveau de l'image.

Traitement de bas niveau
Correction d'histogramme (4)

- Il est possible de retrouver une image plus claire en augmentant le contraste:

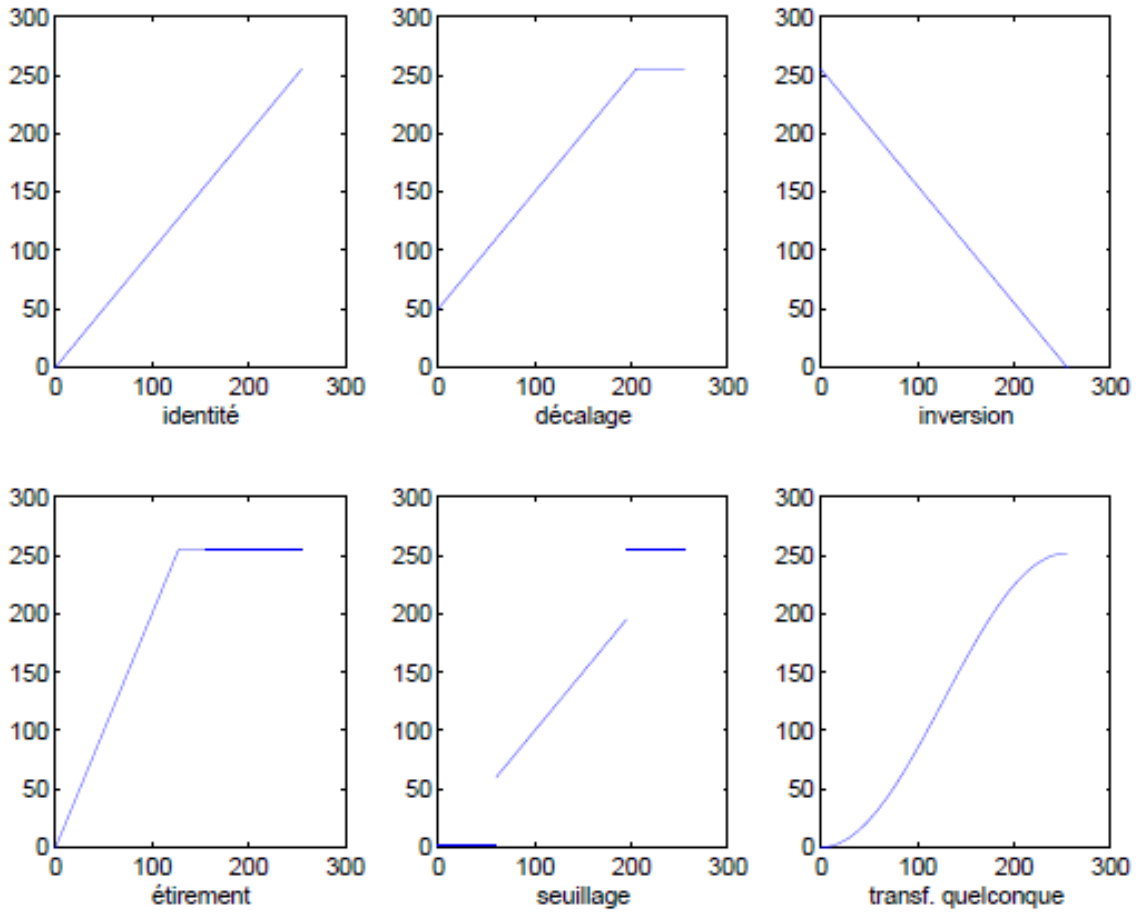


- Il suffit d'effectuer une *transformation* afin d'augmenter le contraste. Dans ce cas-ci, il s'agissait d'un *décalage* et d'un *étirement* qui se traduisent par:

$$I' = (I - 55)\frac{255}{170 - 55}$$

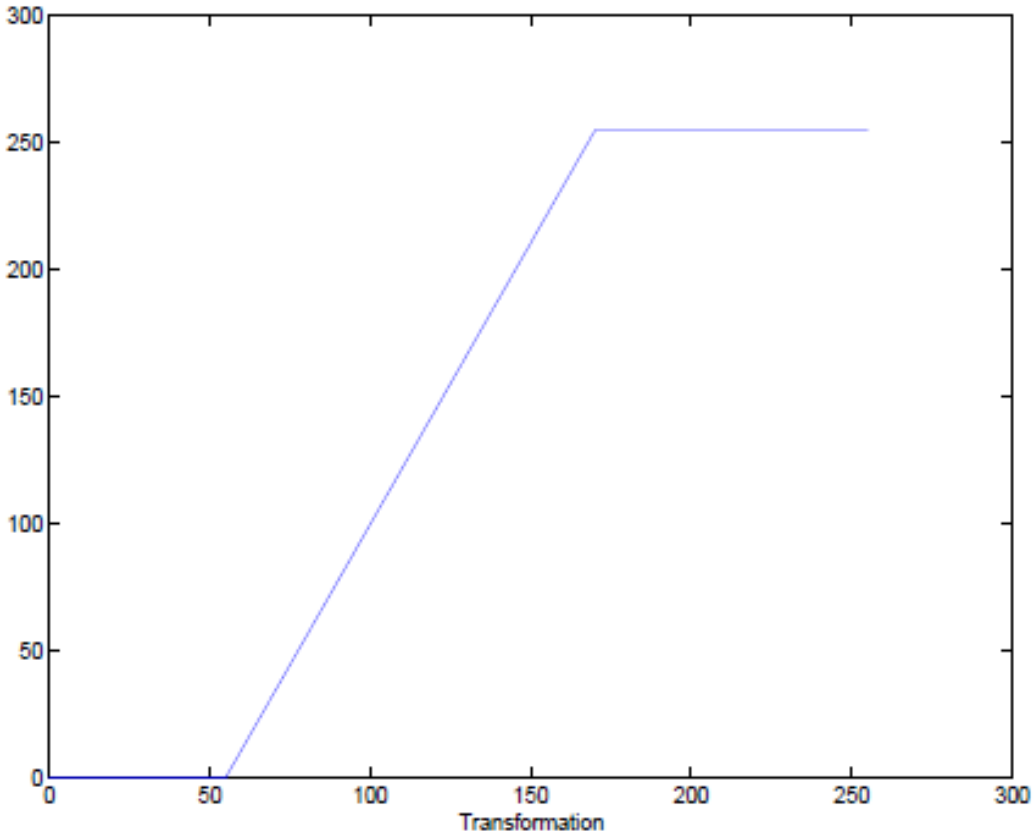
Traitement de bas niveau
Correction d'histogramme (5)

- Autres types de transformation d'histogramme:



Traitement de bas niveau
Correction d'histogramme (6)

- Pour augmenter le contrate de l'image d'oiseau, nous avons utiliser un décalage et un étirement, donc une transformation de ce type:



Traitement de bas niveau
Correction d'histogramme : seuillage(7)

- **Seuillage par histogramme**
  - Considérons cette image et son histogramme:

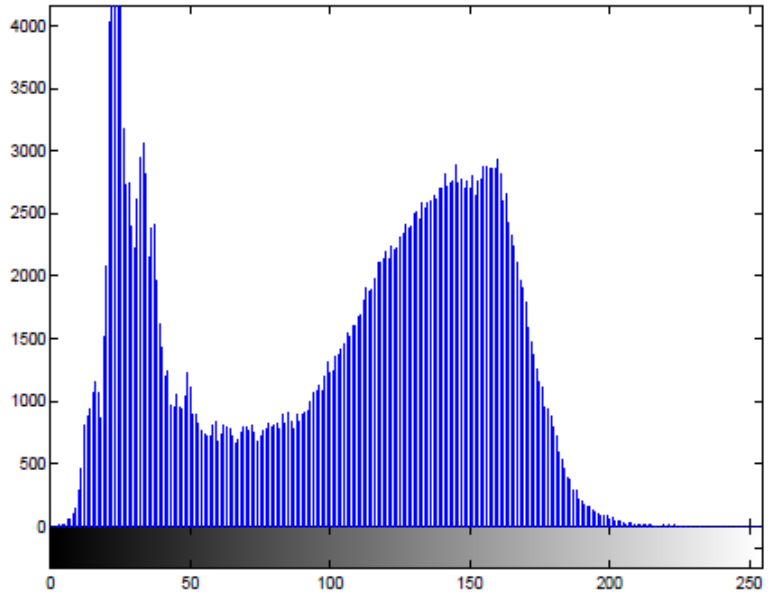


FIG. 2.8 – Histogramme d'une image avant le seuillage

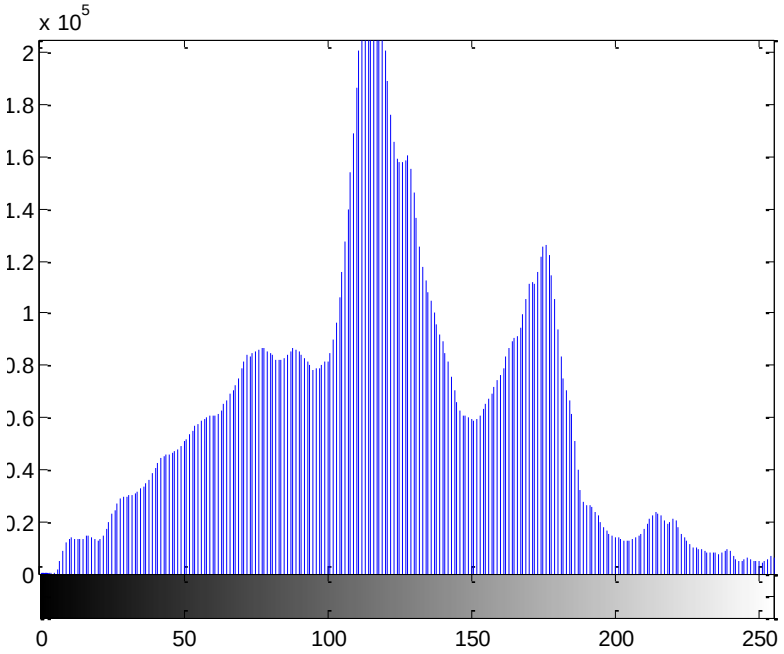
  - On observe que la distribution d'intensité possède deux modes!

Traitement de bas niveau
Correction d'histogramme: seuillage (8)

- Dans certains cas, le seuillage peut donner de très bons résultats:

Traitement de bas niveau
Correction d'histogramme: seuillage(9)

- Alors que d'autres fois, les résultats peuvent être plus compliqués à interpréter:



Traitement de bas niveau
Correction d'histogramme: seuillage(9)

- La morale au niveau de ces démonstrations est que, dans un contexte industriel, vous devez toujours accorder une grande importance à l'éclairage et au type d'éléments que vous voulez percevoir.
  - Un objet éclairé avec la bonne technique sera plus facilement détectable.

- Nous avons vu qu'en observant un histogramme, il était possible de déterminer une valeur seuil permettant de transformer l'image en noir et blanc.
  - **Est-il possible de calculer une valeur seuil automatiquement, sans intervention humaine?**

Traitement de bas niveau
Correction d'histogramme: La méthode d'Otsu

- La méthode d'Otsu permet de trouver une valeur seuil automatiquement.
  - Celle-ci est basée sur l'hypothèse que l'image provient de deux distributions gaussiennes: l'une pour la partie claire et l'autre pour la partie foncée.
  - L'hypothèse est donc que l'histogramme de l'image réelle est une superposition de la distribution gaussienne de la partie pâle avec la distribution de la partie foncée.
    - Cette méthode donnera donc de bons résultats lorsqu'on détecte effectivement dans l'histogramme une distribution *à gauche* et une autre *à droite* avec des moyennes significativement différente.
- **L'idée derrière la méthode d'Otsu est de trouvé un seuil t créant deux groupes dont les variances pondérées sont les plus petites possible.**

Traitement de bas niveau
Correction d'histogramme: La méthode d'Otsu (1)

- ***"L'idée derrière la méthode d'Otsu est de trouvé un seuil t créant deux groupes dont les variances pondérées sont les plus petites possible."***
- Donc on veut minimiser ceci:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

- Avec:

$$q_1(t) = \sum_{i=1}^{t} P(i) \qquad q_2(t) = \sum_{i=t+1}^{I} P(i) = 1 - q_1(t)$$

- On pourrait donc utiliser simplement ces équations et trouver la valeur de t qui minimise la variance $\sigma_w^2(t)$de manière itérative.
  - Il existe tout de même une manière plus efficace.

Traitement de bas niveau
Correction d'histogramme: La méthode d'Otsu (2)

- En effet, puisque Otsu a démontré que minimiser:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

- Revient à maximiser:

$$\sigma_b^2 = q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$$

- Avec $\mu_1(t)$ la moyenne du groupe 1 et $\mu_2(t)$ la moyenne du groupe 2:

$$\mu_1(t) = \frac{\sum_{i=0}^{t} iP(i)}{q_1(t)}$$

$$\mu_2(t) = \frac{\sum_{i=t+1}^{255} iP(i)}{q_2(t)}$$

Traitement de bas niveau
Correction d'histogramme: La méthode d'Otsu (3)

FIG. 2.10 – Image binaire après seuillage : méthode de Otsu

Traitement de bas niveau
Correction d'histogramme: La méthode d'Otsu

- **Exemple d'une application industrielle basée sur le seuillage du niveau de rouge:**

C:\Users\
ge\Desktop\ELE42C

Traitement de bas niveau
Traitement du bruit poivre et sel (1)

- Après un seuillage, l'image sera constitué de 0 et de 1. Que se passe-t-il si l'image contient un trop grand nombre de pixel isolé?
  - On veut par exemple qu'un patron de genre:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{matrix}$$

devienne

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

et que

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

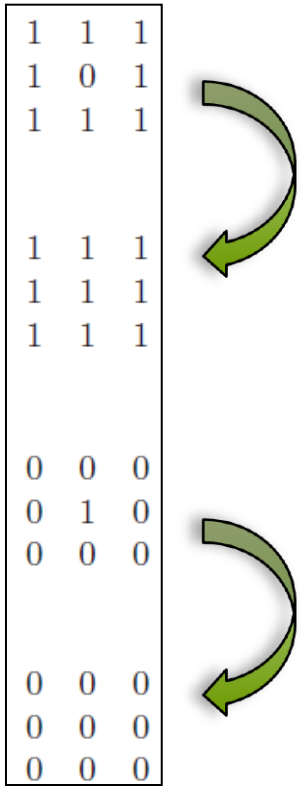
devienne

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Traitement de bas niveau
Traitement du bruit poivre et sel (2)

- On peut réussir ceci à l'aide de masques d'opérateurs logique basés sur les voisins N_4 ou N_8:

$$I'(i,j) = I(i,j) + [I(i,j+1) \cdot I(i-1,j) \cdot I(i+1,j) \cdot I(i,j-1)]$$



Traitement de bas niveau
Application de masques (1)

- Nous venons d'introduire la notion de *masque* pour traiter les images affectée de bruit de type poivre et sel.
  - Un masque est un opérateur sur l'intensité *I* d'un pixel *p* permettant de calculer une nouvelle intensité *I'* en fonction de l'intensité de *p* et de ses voisins. Un masque peut être de dimension *n x n.*
- Forme général d'un masque (ici, 3x3):

$$\begin{aligned} I'(i,j) &= w_1 I(i-1,j+1) + w_2 I(i,j+1) + w_3 I(i+1,j+1) \\ &\quad + w_4 I(i-1,j) + w_5 I(i,j) + w_6 I(i+1,j) \\ &\quad + w_7 I(i-1,j-1) + w_8 I(i,j-1) + w_9 I(i,j+1) \end{aligned}$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Traitement de bas niveau
Application de masques (2)

- **Lissage:**
  - Pour *lisser* une image, on peut utiliser la moyenne du pixel *p* et des pixels dans le voisinnage de *p*. Par exemple, si on utilise $N_8(p)$, on aura $w_i=1/9$.
  - Le lissage a pour avantage d'améliorer les images bruitées, mais le désavantage de rendre floues les arrêtes et dertains détails pointus.
  - De manière similaire, si on considère encore plus de voisins (en utilisant par exemple un masque 5x5) on aura des arrêtes encore plus floues.

Traitement de bas niveau
Application de masques (3)

- **Détection d'arrêtes par gradient :**
  - Pour détecter les arrêtes dans une image, on peut dériver partiellement l'intensité en x et en y en utilisant un filtre approprié:

$$\left(\frac{\partial I}{\partial x}\right) \Rightarrow$$

-1	0	1

  - Cependant, le filtre ci-dessus sera très sensible au bruit, on peut donc utiliser la moyenne des gradients:

$$\left(\frac{\partial I}{\partial x}\right) \Rightarrow$$

-1	0	1
-1	0	1
-1	0	1

ou

-1	0	1
-2	0	2
-1	0	1

  - Remarque:
    - On a qu'a transposer ces gradients pour obtenir l'information dans le sens horizontal: $\left(\frac{\partial I}{\partial y}\right)$

Traitement de bas niveau
Application de masques (4)

- **Le gradient total et sa direction sont donnés par:**

$$|\nabla I| = \sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$$

$$\theta = \text{atan2}\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

- **D'autres masques…**

- **<u>Le Laplacien</u> est défini par:** $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$

  - Il peut aussi être calculé par un masque:

0	1	0
1	-4	1
0	1	0

Traitement de bas niveau
Application de masques (5)

- **D'autres masques…**

- Pour rehausser les contours d'une image, on peut utiliser le masque ci-dessous:

$$\frac{1}{1+\alpha}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

FIG. 2.12 – Figure originale et après application du rehaussement ($\alpha = 0.5$)

Traitement de bas niveau
Exemple (1)

- Soit une image originale avec sa version bruitée:

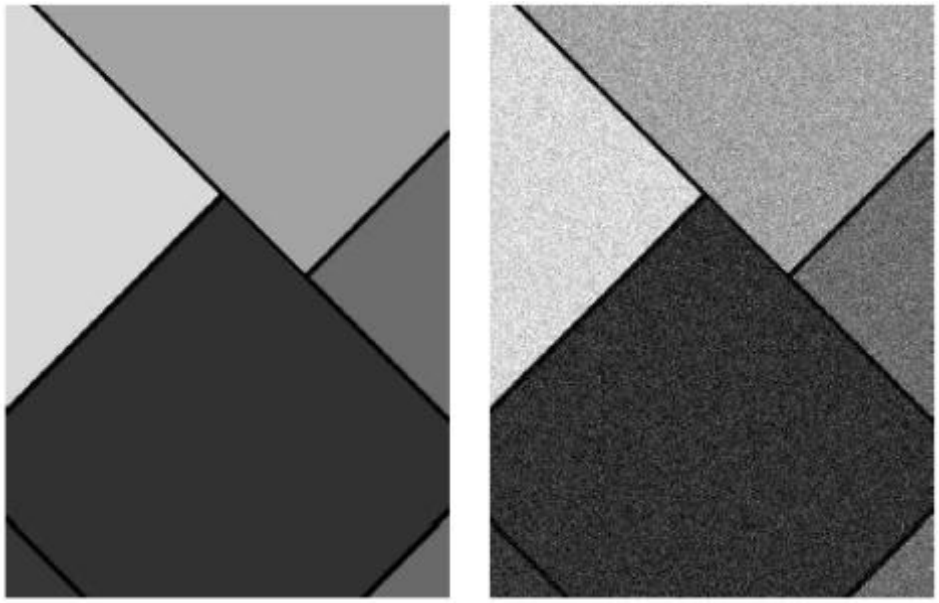


FIG. 2.13 – Images originale et bruitée

- On veut détecter les arrêtes noirs qui séparent les tons de gris…

Traitement de bas niveau
Exemple (2)

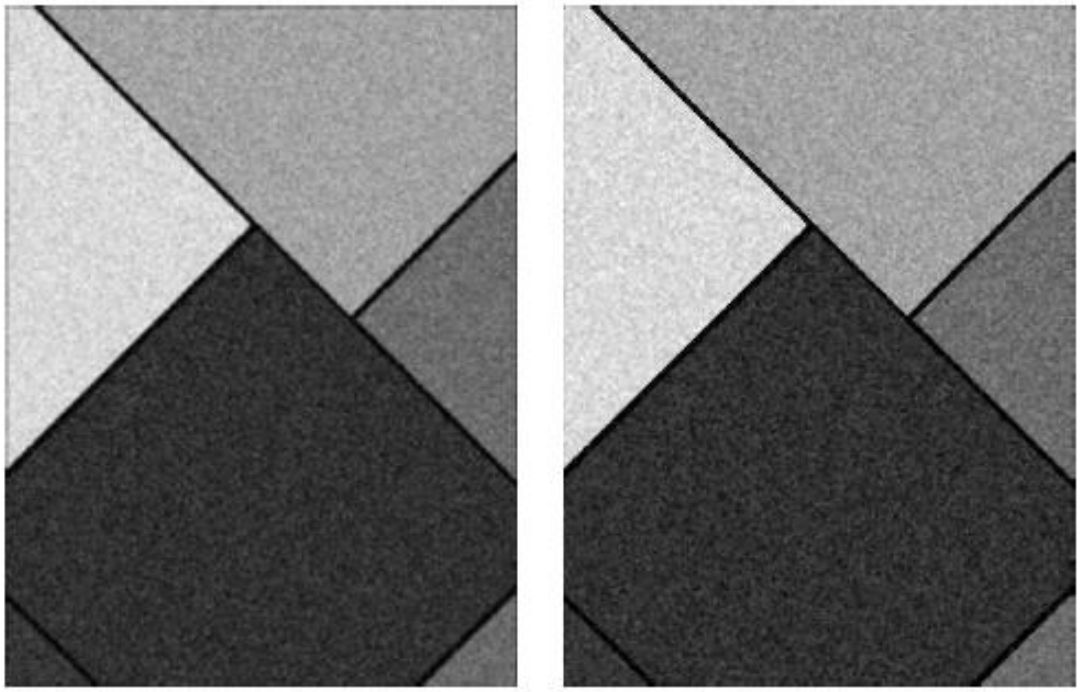


FIG. 2.14 – Images filtrées : moyenne et médiane

Traitement de bas niveau
Exemple (3)

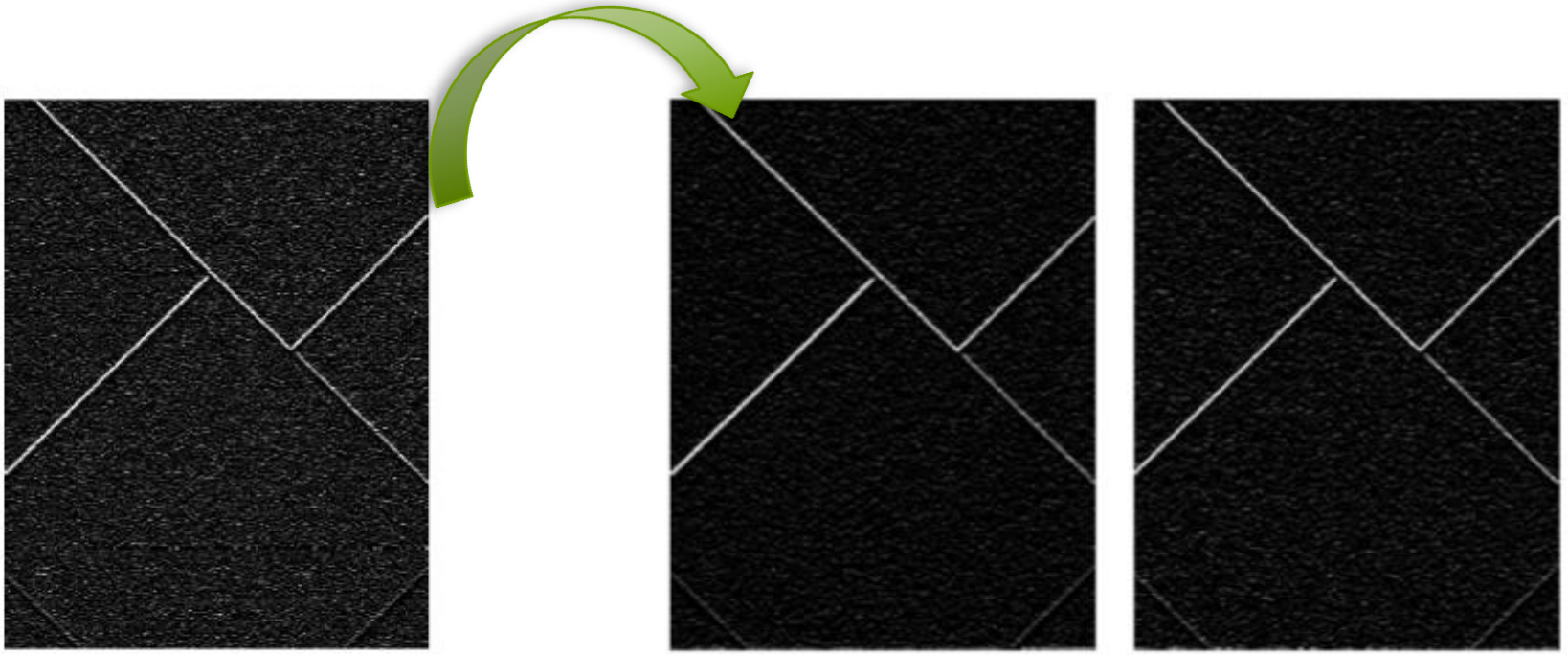


FIG. 2.15 – Gradient de l'image non filtrée

FIG. 2.16 – Gradient des images filtrées : moyenne et médiane

Traitement de bas niveau
Exemple (4)

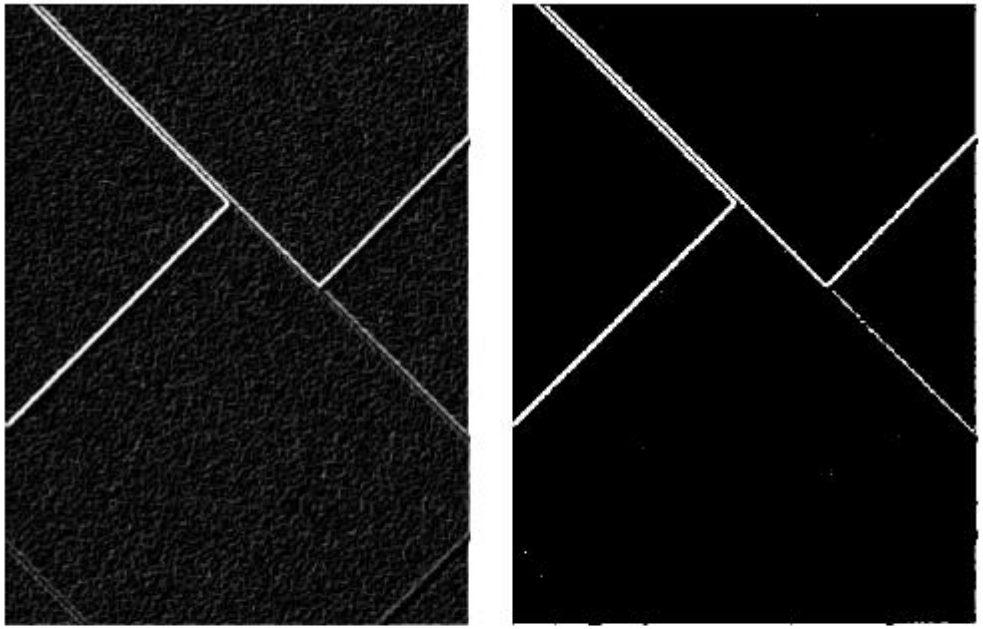


FIG. 2.17 – Norme des gradients de l'image filtrée en moyenne et seuillage des gradients

Traitement de bas niveau
Exemple (5)

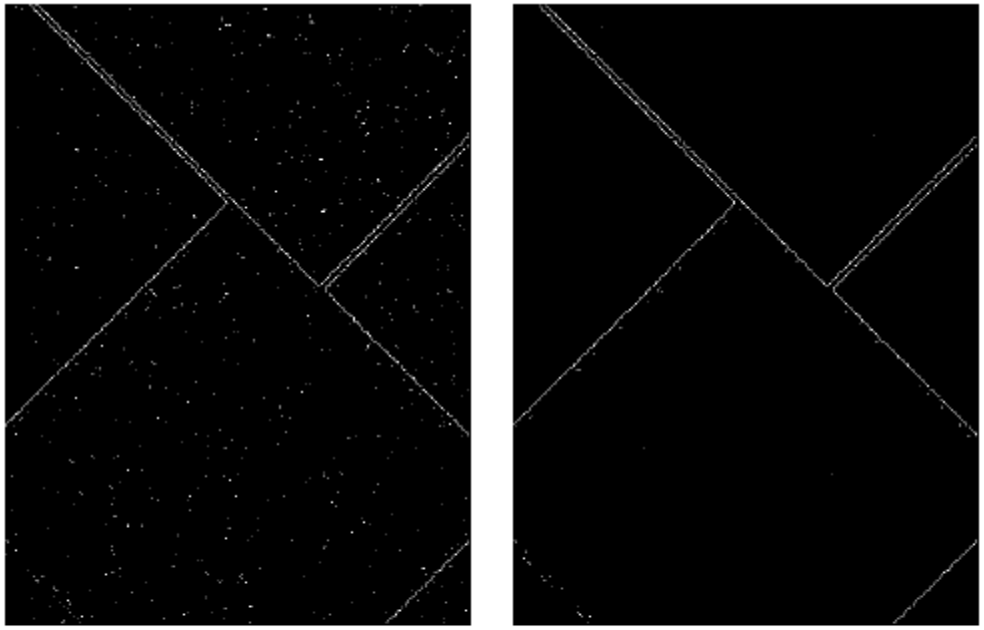


FIG. 2.18 – Détection des contours : fonction MATLAB basée sur les gradients des images non filtrée et filtrée en moyenne

Présentation d'un intérêt : Course automobile

**Basé sur les articles:

-A Novel Framework for Closed-Loop Robotic Motion Simulation – Part 1: Inverse Kinematics Design (P. Robuffo Giordano, C. Masone, J. Tesch, M. Breidt, L. Pollini, ans H. H. Bülthoff), 2010

-A Novel Framework for Closed-Loop Robotic Motion Simulation – Part 2: Motion Cueing Design and Experimental Validation (P. Robuffo Giordano, C. Masone, J. Tesch, M. Breidt, L. Pollini, ans H. H. Bülthoff), 2010

Intérêt: Course Automobile (1)

- Le but de la recherche est d'utiliser les 6 degrés de liberté d'un robot manipulateur Kuka afin de mettre sur pied un simulateur de conduite automobile (ou d'un autre type de véhicule).
  - Le principe:

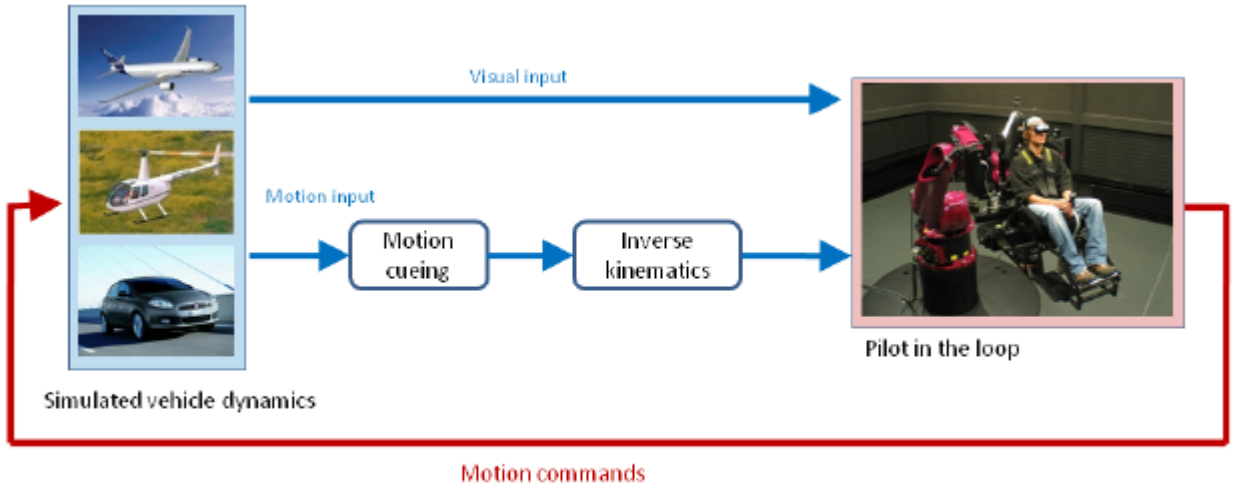
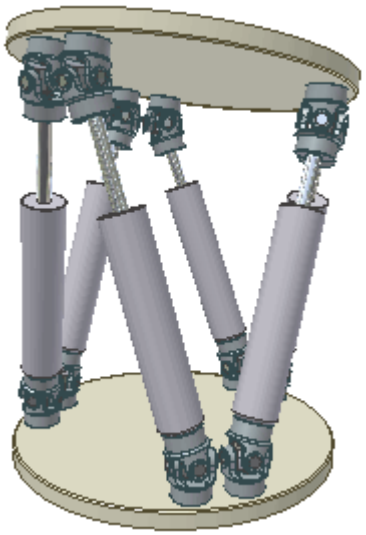


Fig. 2: A block-scheme representation of the system architecture

Intérêt: Course Automobile (2)

- **Autre but:**
  - Voir si un tel manipulateur anthropomorphique n'offrirait pas une meilleur alternative à la classique plate-forme de Stewart:



Intérêt: Course Automobile (3)

- La nacelle du “pilote”:

(a) (b) (c)

Fig. 1: A snapshot of the CyberMotion Simulator setup (a), and some details of the steering wheel and pedals mounted on the cabin (b–c)

Intérêt: Course Automobile (4)

IV. Design of the Inverse Kinematics

A. *Preliminary definitions*

With reference to Fig. 1(a) , let $\mathcal{F}_0 : \{O; \vec{X}_0, \vec{Y}_0, \vec{Z}_0\}$ be a world reference frame fixed to the robot base, with $\vec{Z}_0$ pointing upwards and $(\vec{X}_0, \vec{Y}_0)$ spanning the horizontal plane. A moving reference frame $\mathcal{F}_P : \{O_P; \vec{X}_P, \vec{Y}_P, \vec{Z}_P\}$ is attached to the the pilot's head (supposed fixed to the cabin) and has its axes aligned with the pilot's forward/left/upward direction, respectively.

Furthermore, let R_P be the rotation matrix from frame $\mathcal{F}_0$ to frame $\mathcal{F}_P$, and $\eta = [\rho\ \theta\ \psi]^T \in \mathbb{R}^3$ the usual set of roll-pitch-yaw Euler angles parameterizing R_P. Let also $p = [x\ y\ z]^T \in \mathbb{R}^3$ represent the coordinates of O_P in $\mathcal{F}_0$. With these settings, we will call $J_{CE}(q) \in \mathbb{R}^{6\times 6}$ the Jacobian matrix representing the mapping from joint velocities $\dot{q}$ to Cartesian/Euler velocities

$$\begin{bmatrix} \dot{p} \\ \dot{\eta} \end{bmatrix} = J_{CE}(q)\dot{q}. \qquad (2)$$

Derivation of $J_{CE}(q)$ follows from any standard robotics textbook, see, e.g., [13].

1) Singularities: singularities occur at those configurations $\bar{q}$ where the task Jacobian $J(\bar{q})$ loses rank, i.e., in our case when rank $J(\bar{q}) < 6$. At a singular configuration it is impossible to generate task velocities in certain directions, namely those directions associated to the zero singular values of J. It is also well-known that, apart from the loss of mobility, any method based on the (pseudo-)inversion of J will become numerically unstable close to a singularity, yielding unbounded joint velocity commands as $q(t)$ approaches $\bar{q}$.

In this respect, a convenient way to deal with the occurrence of singularities is to resort to a Task Priority (TP) inversion scheme [20]. The idea is to divide the main task into several subtasks with different priorities. Away from singularities the whole task is realized whereas, whenever J is rank deficient, the lowest priority tasks are automatically relaxed while still correctly executing those with highest priority.

Intérêt: Course Automobile (5)

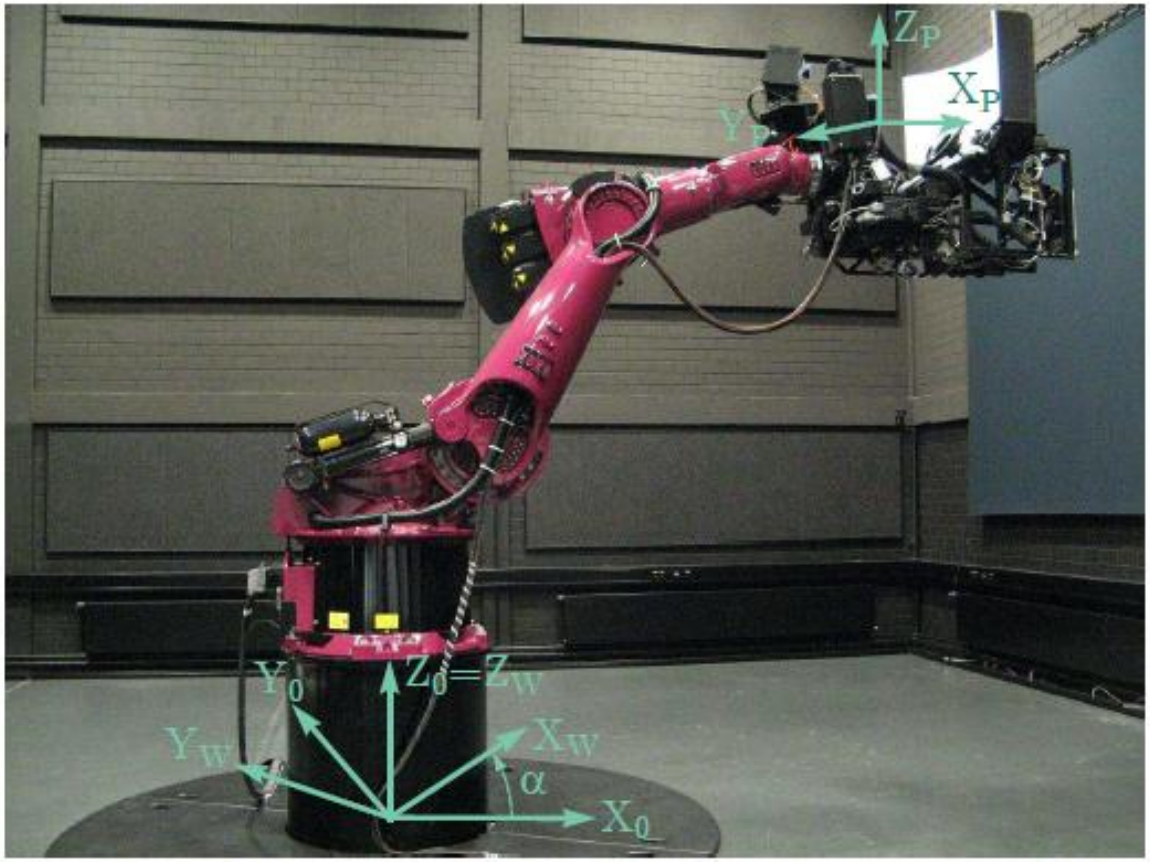


Fig. 1: A snapshot of the CyberMotion Simulator setup with some relevant reference frames

Intérêt: Course Automobile (6)

C:\Users\
ge\Desktop\ELE42C

Références

- [1] *Absolute Beginner's Guide to Building Robots,* Gareth Branwyn, 2003
- [2] *http://spectrum.ieee.org/automaton/robotics/robotics-software/10_stats_you_should_know_about_robots Notes de cours (ELE3202) – Richard Gourdeau & John Thistle*
- [3] *http://www.geekologie.com/2008/12/thats-it-im-moving-robotic-sta.php*
- [4] *Robot Modeling and Control, Mark W. Spong et al.,2006.*
- [5] Notes de cours (Manipulateurs) - ELE4203, Richard Gourdeau, juillet 2012.