

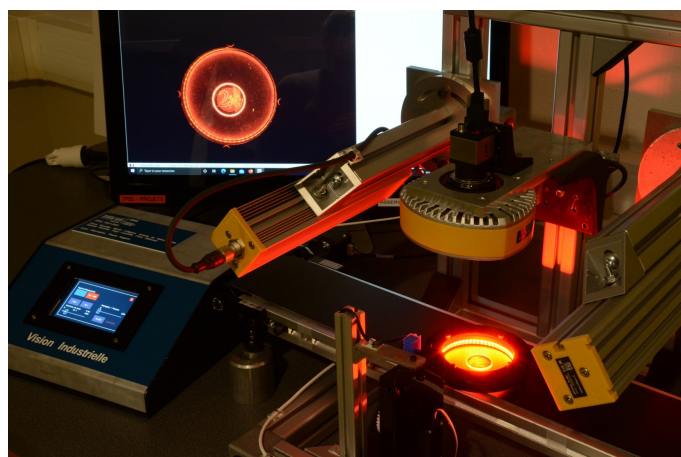
INTERFAÇAGE NUMÉRIQUE

Travaux Pratiques

Semestre 6

Chaine d'acquisition d'une image Manipulation d'images

2 séances



Ce sujet est disponible au format électronique sur le site du LEnSE - <https://lense.institutoptique.fr/> dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc Caméra.

Chaîne d'acquisition d'une image et manipulation d'images avec OpenCV

Objectifs du bloc

L'objectif principal de ce bloc est de découvrir les **différents paramètres impactant la qualité d'une prise d'image par une caméra** dans un environnement industriel ainsi que la **manipulation d'images numériques**, à l'aide de la bibliothèque **OpenCV**.

On s'intéressera à l'impact du **temps d'intégration**, de la **résolution** de la caméra et de l'**éclairage** (couleur et intensité) de la scène et des objets sur l'image résultante.

À l'issue des séances de TP concernant le bloc sur les caméras industrielles et la manipulation d'images, les étudiant-es seront capables de :

- paramétrer le temps d'exposition en fonction de l'intensité lumineuse pour **obtenir une image d'une qualité exploitable**
- utiliser des outils d'analyse (histogramme notamment) pour **améliorer numériquement** la qualité d'une image (contraste, luminosité...)
- **manipuler une image numérique** à l'aide de la bibliothèque *OpenCV*

Cette étude sera complétée par un TP plus détaillé sur les bruits associés à l'utilisation des caméras CMOS en 2ème année du cycle ingénieur à Palaiseau.

Des cours et des projets autour du traitement d'images sont également proposés dans les prochaines années de formation, quelque soit le site. Ces deux séances sont une introduction à ces modules plus avancés.

*Le bloc **Images et OpenCV** (facultatif) de ce module permet d'aller également plus loin dans le pré-traitement des images et leur manipulation.*

Ressources

— ??

Déroulement du bloc

Séance 1 - Modélisation primaire d'une chaîne d'acquisition

Etape 0 - 30 min Prendre en main l'interface de pilotage

Etape 1 - 30 min Décrire le rôle des principales caractéristiques d'une caméra CMOS (temps d'intégration, black level...)

Etape 2 - 30 min Analyser la pertinence de l'éclairage sur une scène à l'aide de l'histogramme d'une image

Etape 3 - 30 min Analyser l'impact du choix de la couleur de l'éclairage en fonction des caractéristiques d'un objet

Etape 4 - 30 min Analyser l'impact de l'échantillonnage et la quantification sur les images

Etape 5 - 30 min Réponse de la caméra

Etape 6 - 60 min Analyser l'impact des outils de base de la manipulation d'images (*Contraste, luminosité, seuillage et filtrage*)

Séance 2 - Manipulation d'images

Etape 1 - 20 min Ouvrir une image sous OpenCV (niveau de gris et couleur) et extraire les informations utiles de l'image

Etape 2 - 20 min Calculer l'histogramme d'une image et l'afficher

Etape 3 - 20 min Améliorer numériquement la qualité d'une image : contraste, luminosité...

Etape 4 - 90 min Appliquer un filtre moyenneur (*Gaussian blur*) sur une image et analyser l'impact du choix du noyau

Etape 5 - 60 min Appliquer un filtre passe-haut (*Roberts, Sobel*) sur une image et analyser l'impact du choix du noyau

Etape 6 - 30 min Appliquer un filtre par l'intermédiaire de la transformée de Fourier

Séance 1 / Modélisation primaire d'une chaîne d'acquisition

Ce bloc de travaux pratiques utilise un **banc de vision industrielle** avec une lampe de type Effi-Ring RGB, une caméra Basler et une interface développée en **Python** (PyQt6) et qui utilise des fonctionnalités de la bibliothèque **OpenCV**.

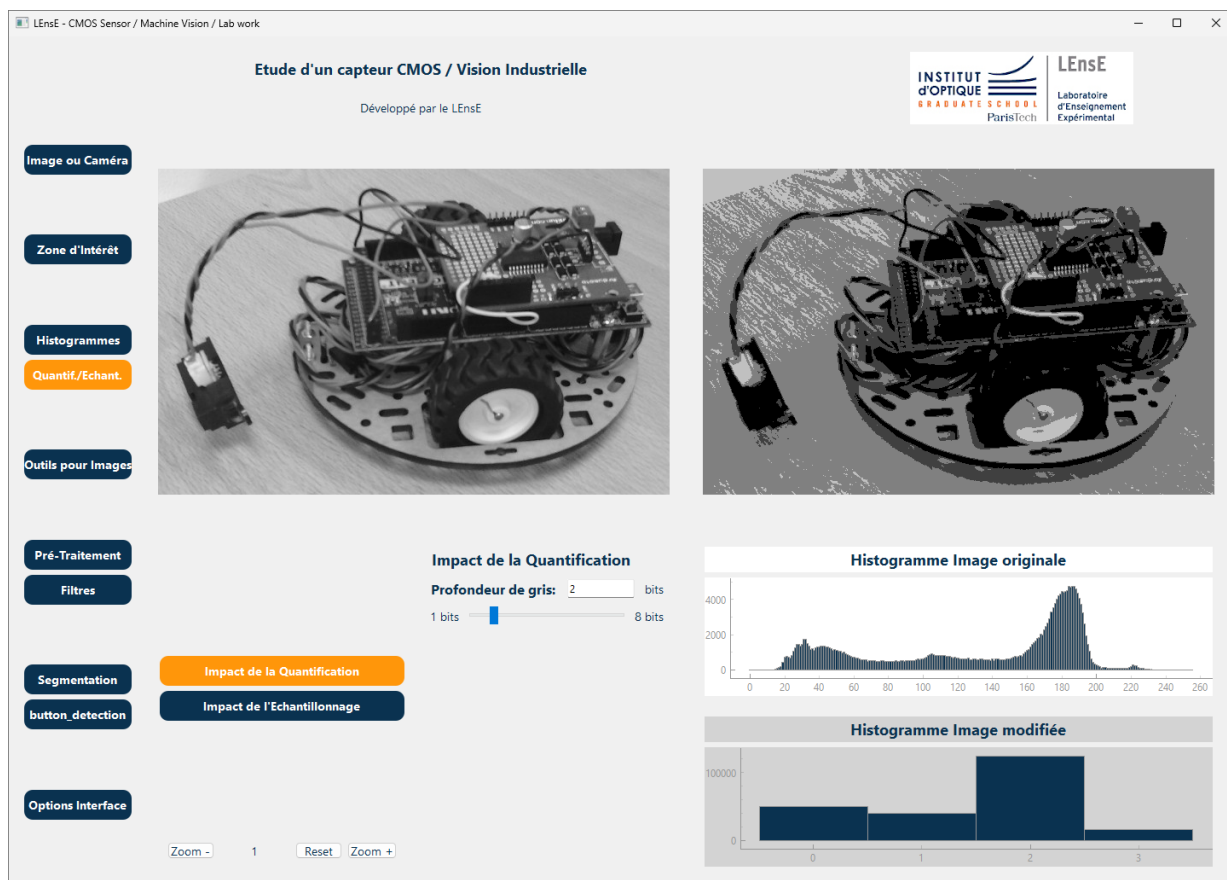
Les documentations de la caméra et de l'éclairage sont disponibles aux adresses suivantes :

- Basler **a2A 1920 - 160ucBAS** : <https://docs.baslerweb.com/a2a1920-160ucbas#specifications>
- Effi-Ring : <https://www.ffmpeg.com/fr/produits/annuaire/effi-ring>

Prise en main de l'interface

Temps conseillé : 30 min

- **M** Lancer l'application **CMOS_Machine_Vision** depuis ...
 (pour l'instant dernière version officielle sur le github suivant :
<https://github.com/IOGS-LEnSE-ressources/camera-gui>
 - répertoire applis/CMOS-MachineVision_v3)



→ M Ouvrir la première caméra disponible dans l'onglet IMAGE OU CAMÉRA / SÉLECTIONNER UNE CAMÉRA / OUVRIR LA PREMIÈRE CAMÉRA.

Eclairage et zone d'intérêt

- M Allumer l'éclairage annulaire du banc (trois couleurs).
- Q Quelle couleur d'éclairage obtient-on ?
- M Placer un objet (un cube de couleur par exemple) dans le champ de la caméra.
- M Ajuster la zone d'intérêt (ou *Area of Interest* - AOI) à l'aide de l'onglet ZONE D'INTÉRÊT pour ne sélectionner qu'une partie de l'image autour de l'objet.
- Q Que pouvez-vous dire des deux histogrammes affichés ?

Décrire le rôle des principales caractéristiques d'une caméra CMOS

Temps conseillé : 30 min

Temps d'intégration

- M Dans l'onglet IMAGE OU CAMÉRA, sélectionner un *Black Level* de 0. Modifier le temps d'intégration de la caméra.
- Q Que constatez-vous sur l'histogramme de l'image ? Sur la moyenne et l'écart-type de la répartition de la luminosité des pixels ?
- M Placer un cache devant la caméra (ou Prévoir une caméra annexe avec cache - pour éviter d'enlever l'objectif et mettre un cache...) pour vous placer dans l'obscurité.
- Q La répartition de la luminosité perçue par chaque pixel est-elle uniforme ? Que pouvez-vous en conclure ?

Black Level

- M Dans l'obscurité, relever l'histogramme de l'image ainsi que la moyenne et l'écart-type de la répartition de la luminosité des pixels pour différentes valeurs du *black level* : 0, 10, 20, 50.
- Q Que peut-on conclure de l'intérêt du black level ?

Analyser la pertinence de l'éclairage d'une scène

Temps conseillé : 30 min

Utilisation de l'histogramme et du seuillage par exemple pour définir le bon éclairage permettant de séparer l'objet du fond ?
Trouver des échantillons pertinents.

Analyser l'impact du choix de la couleur de l'éclairage

Temps conseillé : 30 min

Caméra monochrome : quelle couleur d'éclairage pour détecter la couleur d'un objet ? (premier pas vers colorimétrie/photométrie)

Trouver des objets qui ont des comportements similaires en GRAY avec des couleurs différentes.

Analyser l'impact de l'échantillonnage et la quantification sur les images

Temps conseillé : 30 min

Matériel

Objets de tailles différentes, de formes différentes...

Mire papier dans le champ de la caméra pour faire des mesures ?

Trouver un échantillon (ou une image préenregistrée) permettant par sous-échantillonnage d'obtenir un autre motif (Moiré)

Questions

- > Quelle est l'influence du choix d'un pas de quantification ?
- > Quelle est l'influence du choix d'un pas d'échantillonnage ?

Réponse de la caméra

Temps conseillé : 30 min

A partir des spectres des sources RGB, essayer (sur un même objet - ayant des propriétés quasi équivalentes en réflexion quelque soit la longueur d'onde du visible) de caractériser la courbe de réponse de la caméra ?

Avec une lumière blanche et des filtres ?

Analyser l'impact des outils de base de la manipulation d'images

Temps conseillé : 60 min

Via l'interface, jouer sur *Contraste*, *luminosité*, *seuillage* et *filtrage* et voir l'effet sur l'histogramme et sur l'image.

Séance 2 / Manipulation d'images (OpenCV) et filtrage

Lors de cette séance, vous devrez écrire vos propres scripts en **Python** (avec l'IDE PyCharm par exemple) permettant de réaliser des opérations de base de manipulation d'images, à l'aide notamment de la célèbre bibliothèque **OpenCV**.

Ressources

Un tutoriel sur les bases d'OpenCV est disponible à l'adresse suivante :

<https://iogs-lense-training.github.io/image-processing/>

Un **kit d'images** est disponible sur le site du LensE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc Images et OpenCV / Kit d'images*.

Diapos : SC19 - Image Processing

Ouvrir une image sous OpenCV

Temps conseillé : 20 min

Notions : *Open an image - Display an image*

→ M Créer un nouveau projet sous PyCharm.

→ M Impoter la bibliothèque **OpenCV2** (*cv2*).

Pour la suite, il est nécessaire de placer **tous les fichiers dans le même dossier** : votre script Python, les différentes images et les bibliothèques de fonction (fichier *images_manipulation.py*)

→ M Ouvrir l'image *robot.jpg* du kit d'images fourni, au format RGB.

→ Q Quelle est la taille de l'image ? Quel est le type de données d'un élément ?

→ M Ouvrir l'image *robot.jpg* du kit d'images fourni, en niveau de gris.

→ Q Quelle est la taille de l'image ? Quel est le type de données d'un élément ?

Calculer l'histogramme d'une image et l'afficher

Temps conseillé : 20 min

Notions : *Calculate the histogram*

→ M Calculer l'histogramme de l'image précédente et l'afficher.

Il peut être intéressant de **créer une fonction qui affiche automatiquement l'histogramme d'une image à partir de ses données**. Elle sera très utile dans la suite du TP pour voir l'impact des effets appliqués sur les images.

Améliorer numériquement la qualité d'une image

Temps conseillé : 20 min

Notions : *Enhance Contrast/Brightness*

Pour modifier le contraste et la luminosité d'une image, il faut appliquer une transformation linéaire à **chaque pixel** de l'image pouvant être exprimé mathématiquement comme suit :

$$P_{new} = \alpha \cdot P_{old} + \beta$$

où α est le facteur de contraste. Une valeur supérieure à 1 augmente le contraste, tandis qu'une valeur entre 0 et 1 le réduit.

β est l'offset de luminosité. Une valeur positive rend l'image plus lumineuse, tandis qu'une valeur négative l'assombrit.

On utilisera la fonction `cv2.convertScaleAbs()` pour modifier le contraste et la luminosité de l'image.

→ M Ouvrir l'image *robot.jpg* du kit d'images fourni, en niveau de gris.

→ M Modifier le contraste de l'image et comparer les histogrammes de l'image originale et de la version modifiée pour différente valeur de α .

→ M Modifier la luminosité de l'image et comparer les histogrammes de l'image originale et de la version modifiée pour différente valeur de β .

Appliquer un filtre moyenneur sur une image

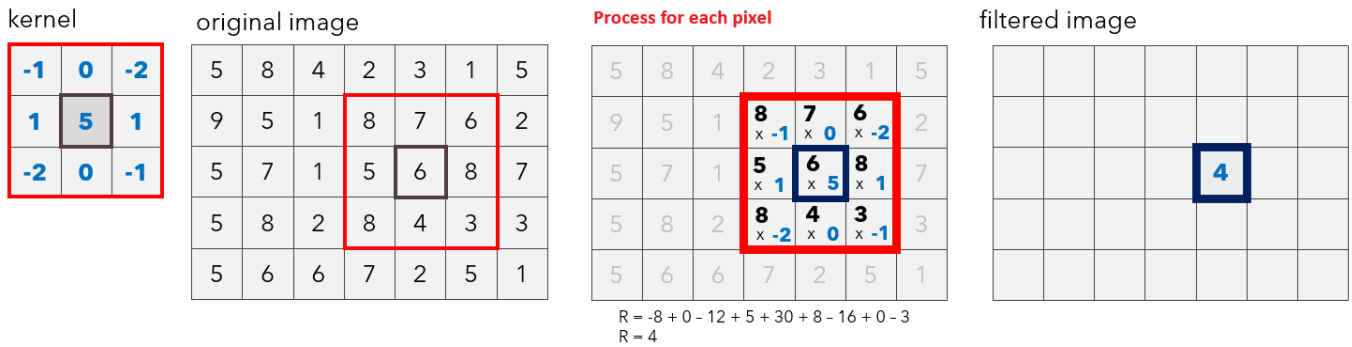
Temps conseillé : 90 min

La transformation précédente ne prend pas en compte les pixels voisins. Il est pourtant intéressant dans de nombreuses situations de capturer des relations spatiales entre les pixels.

Les filtres prenant en compte les pixels voisins exploitent les relations locales dans une image, ce qui est crucial pour des tâches comme la suppression de bruit, la détection de contours, l'extraction de caractéristiques, et l'amélioration de la qualité visuelle. Travailler sur des pixels isolés limite l'analyse à des informations ponctuelles, tandis que considérer les voisins permet une compréhension plus riche et contextuelle de l'image.

Éléments structurants ou noyau

Un **élément structurant** (ou noyau) est une petite matrice (généralement de taille et de forme prédéfinies, comme un carré, un disque, une ligne, etc.) qui sert de sonde pour inspecter et modifier les pixels d'une image.



Les éléments structurants jouent un rôle clé en traitement d'image, notamment dans les opérations de morphologie mathématique. Ces opérations sont principalement utilisées pour analyser et traiter des images binaires ou en niveaux de gris en modifiant leurs formes ou en extrayant des structures spécifiques.

Filtre moyenneur gaussien

Notions : *Blur with OpenCV*

- M Créer un nouveau projet sous PyCharm.
- M Impoter la bibliothèque **OpenCV2** (*cv2*).
- M Ouvrir l'image *bricks2.jpg* du kit d'images fourni, en niveau de gris.
- M Appliquer un filtre gaussien (fonction *cv2.GaussianBlur()*) sur l'image précédente, de taille 5 x 5 pixels.
- M Afficher les deux images pour les comparer.
- Q Comment peut-on montrer de manière objective les modifications apportées à l'image ?
- M Mettre en oeuvre la méthode proposée.

On se propose d'utiliser la fonction *compare_blur_fft()* fournie dans le fichier *images_manipulation.py* pour comparer l'effet.

Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc Images et OpenCV / Répertoire vers codes à tester*.

- M Tester cette fonction sur l'image précédente.
- Q Etudier cette fonction. Quelles méthodes sont utilisées pour comparer les images ?
- Q Quelle est la fonction réalisée par le filtre précédent ?

On cherche à présent à voir l'impact du noyau sur l'image finale.

→ M Tester la fonction précédente avec des noyaux de taille différente et comparer les résultats à la fois sur l'image obtenue mais également sur la transformée de Fourier.

- Q Que pouvez-vous conclure sur l'impact de la taille du noyau ?

Bruit sur une image

On se propose d'étudier la fonction *generate_gaussian_noise_image()* fournie dans le fichier *images_manipulation.py*.

- M Tester l'exemple fourni dans le fichier *noise_test1.py*.

→ Q Comment vérifier la distribution du bruit généré par cette fonction ?

On se propose d'étudier la fonction `generate_uniform_noise_image()` fournie dans le fichier `images_manipulation`

→ M Tester l'exemple fourni dans le fichier `noise_test2.py`.

→ Q La distribution du bruit généré par cette fonction est-elle uniforme ?

→ M A l'aide de la fonction `generate_gaussian_noise_image_percent()`, générer un bruit gaussien de moyenne 30 et d'écart-type 20 sur 10% de l'image `robot.jpg` ouverte précédemment en nuance de gris. Visualiser le résultat.

*Il peut-être nécessaire de normaliser les images bruitées afin que les **données entières** de chaque pixel soient comprises **entre 0 et 255**, afin que les images puissent être affichées correctement.*

→ M Tester la fonction `compare_blur_fft()` fournie dans le fichier `images_manipulation.py` et comparer les résultats à la fois sur l'image obtenue mais également sur la transformée de Fourier.

Appliquer un filtre passe-haut sur une image

Temps conseillé : 60 min

Le **filtre moyennneur** précédent permet de conserver les **éléments à basse fréquence spatiale** dans l'image. C'est une méthode intéressante pour supprimer des bruits ponctuels (des éléments isolés et donc "rapides"). Il est également possible en choisissant un autre élément structurant de réaliser l'opération complémentaire qui supprime le fond continu et ne conserve que les transitions de fréquence spatiale élevée (bords d'un objet par exemple).

Il est possible d'utiliser la fonction `cv2.filter2D()` pour appliquer un noyau particulier sur une image.

Opérateur de Roberts

L'opérateur de Roberts est l'un des premiers filtres de **détection de contours**. Il repose sur la convolution avec deux petits noyaux 2x2, conçus pour approximer les dérivées en diagonale de l'image.

Les noyaux de convolution sont les suivants :

$$K_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}, \quad K_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}.$$

Il est alors possible de calculer l'amplitude du gradient par l'opération suivante :

$$\text{Amplitude} = \sqrt{G_x^2 + G_y^2}$$

où G_x est le résultat de la convolution de l'image par le noyau K_x et G_y le résultat de la convolution de l'image par le noyau K_y .

Une forte amplitude indique un contour ou un bord marqué. Une faible amplitude indique une région où l'intensité est relativement constante.

→ M Ecrire un script qui permet d'appliquer cette opération sur l'image `bricks2.jpg` du kit d'images fourni, en niveau de gris.

→ M Visualiser l'image originale, le résultat du filtrage selon X et le résultat du filtrage selon Y.

→ Q Que pouvez-vous conclure sur l'effet de ce filtre ?

Opérateur de Sobel

L'opérateur de Sobel permet de réaliser une opération similaire à celui de Roberts, mais en étant moins sensible aux bruits dans l'image, puisqu'il se base sur un noyau plus large et ainsi lisse l'image dans la direction perpendiculaire au gradient mesuré.

Les noyaux de convolution sont les suivants :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

De la même façon que précédemment, on peut calculer l'amplitude du gradient par l'opération suivante :

$$\text{Amplitude} = \sqrt{G_x^2 + G_y^2}$$

où G_x est le résultat de la convolution de l'image par le noyau K_x et G_y le résultat de la convolution de l'image par le noyau K_y .

- M Ecrire un script qui permet d'appliquer cette opération sur l'image *bricks2.jpg* du kit d'images fourni, en niveau de gris.
- M Visualiser l'image originale, le résultat du filtrage selon X et le résultat du filtrage selon Y.
- Q Que pouvez-vous conclure sur l'effet de ce filtre ?

Appliquer un filtre par l'intermédiaire de la transformée de Fourier

Temps conseillé : 30 min

On s'intéresse ici à la **transformée de Fourier discrète en deux dimensions** permettant de représenter l'image dans le domaine fréquentiel, plutôt que dans le domaine spatial.

- M Ouvrir l'image *robot.jpg* du kit d'images fourni, en niveau de gris.
- M Calculer la transformée de Fourier discrète de cette image à l'aide de la fonction *fft2()* de la bibliothèque **Numpy** / **FFT**. Afficher l'image et sa transformée de Fourier. *Pensez à utiliser la fonction *fftshift*...*

On se propose d'utiliser la fonction *circular_mask()* fournie dans le fichier *images_manipulation.py* pour appliquer un masque sur la transformée de Fourier de l'image.

Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interface Numérique S6 / Bloc Images et OpenCV / Répertoire vers codes à tester*.

- M Appliquer un masque circulaire sur la transformée de Fourier de l'image, à l'aide de la fonction *circular_mask()*. Afficher le résultat.
- M Calculer l'image résultante à l'aide de la fonction *ifft2()* de la bibliothèque **Numpy** / **FFT**. Afficher l'image résultante et la comparer à l'image originale.
- M Ecrire et tester une fonction permettant de réaliser un masquage rectangulaire sur une image, dont on pourra préciser la largeur et la longueur, ainsi que le barycentre du rectangle (point d'intersection des diagonales).
- Q Que se passe-t-il lorsque vous appliquez un masque rectangulaire (dont la largeur et la longueur sont différentes) sur une image ? Qu'en est-il d'un masque circulaire ?

INTERFAÇAGE NUMÉRIQUE

Travaux Pratiques

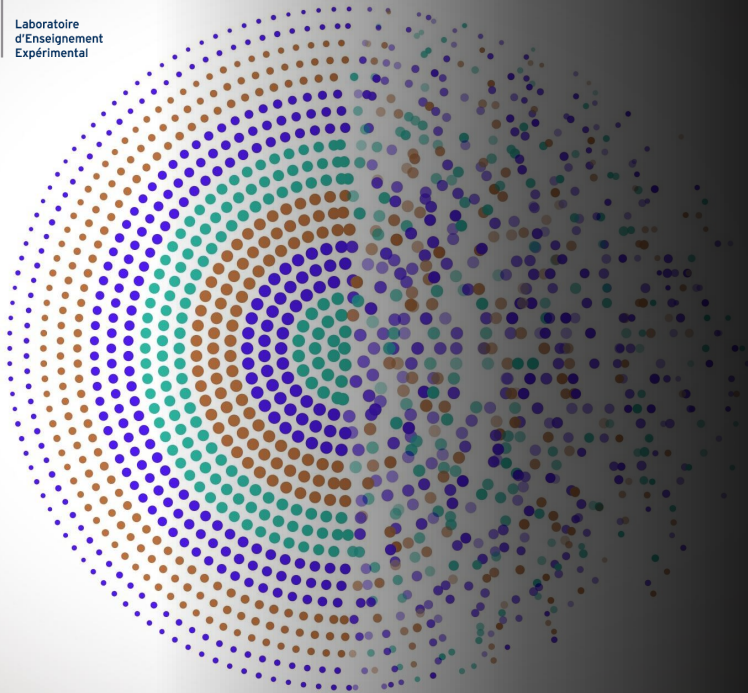
Semestre 6

Ressources

Bloc Caméra

Liste des ressources

— [Camera and sensor / Key concepts](#)



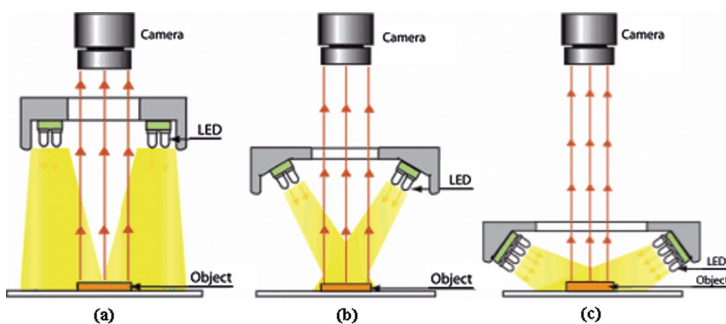
Cameras for Machine Vision

Institut d'Optique – Engineers Training
Semester 6 – Digital Interface

Julien VILLEMEJANE

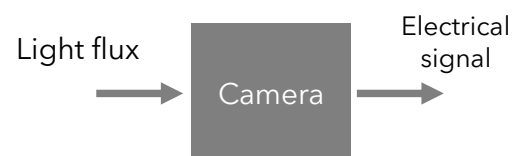
Cameras and Interfaces

Camera in a machine vision system



Camera

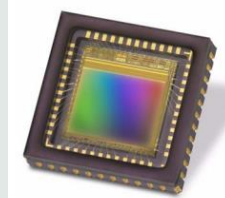
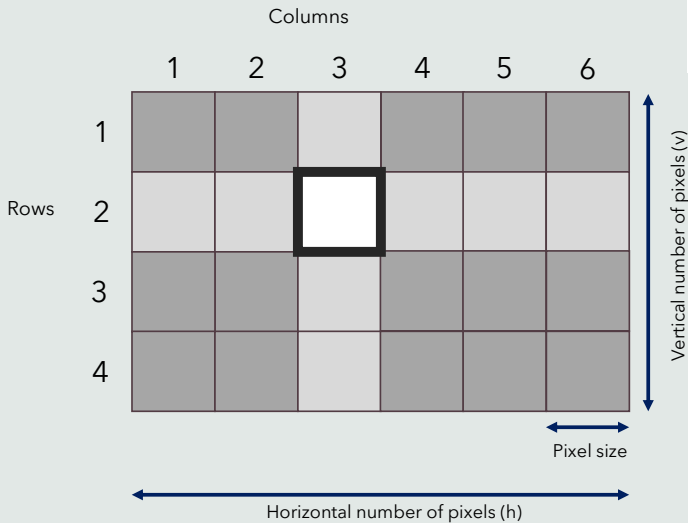
Device that transforms a **light flux** into a **measurable electrical signal**



Dong, Jing-Tao & Lu, rs & Shi, Yan-Qiong & Xia, Rui-Xue & Li, Qi & Xu, Yan. (2011). Optical design of color light-emitting diode ring light for machine vision inspection. Optical Engineering - OPT ENG. 50. 10.1117/1.3567053.

Cameras and Interfaces

Camera : array of small sensors



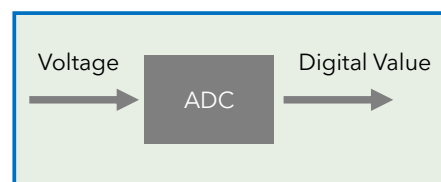
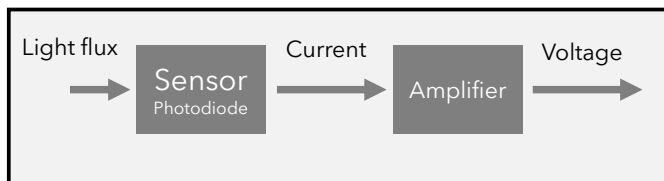
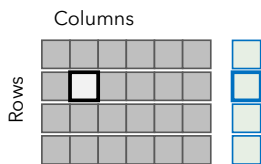
Capteur

Type de capteur	CMOS Couleur
Mode d'obturation	Global / Rolling / Global Start
Caractéristique du capteur	Linéaire
Méthode de lecture du capteur	Progressive scan
Classe de pixels	1.3 MP
Résolution	1,31 Mpx
Résolution (h x v)	1280 x 1024 Pixel
Rapport hauteur/largeur	5:4
CAN	10 bit
Profondeur des couleurs (caméra)	8 bit
Classe de capteur optique	1/1,8"
Surface optique	6,784 mm x 5,427 mm)
Diagonale du capteur optique	8,69 mm (1/1,84")
Taille de pixel	5,3 µm
Fabricant	e2v
Désignation du capteur	EV76C560ACT
Amplification (complet/RVB)	4x/4x

Resolution
Size / Form factor

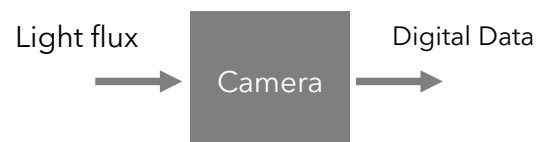
Cameras and Interfaces

From analog signal to digital data

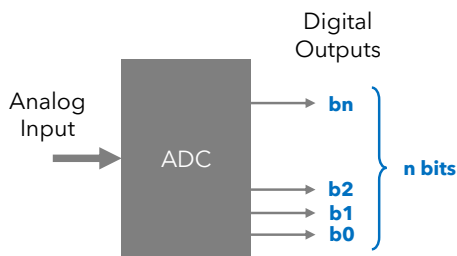
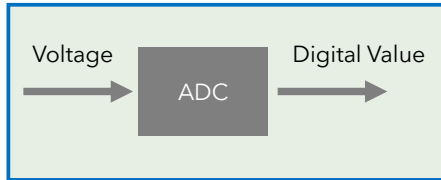


Digital Camera

Device that transforms an array of **light flux sensors** into **digital data** called pixels



How an Analog to Digital Converter works ?

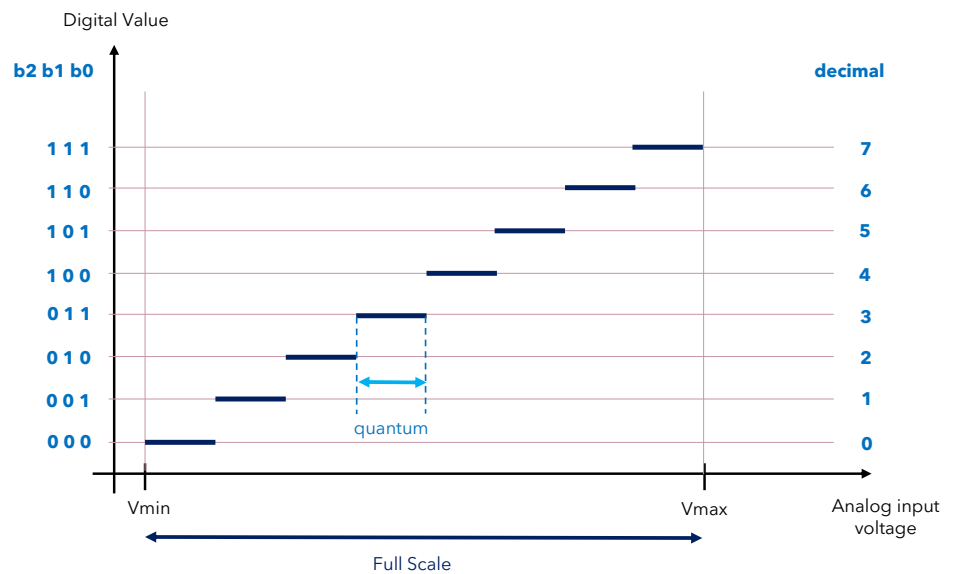


Each bit can have one of two values: **0** or **1**.

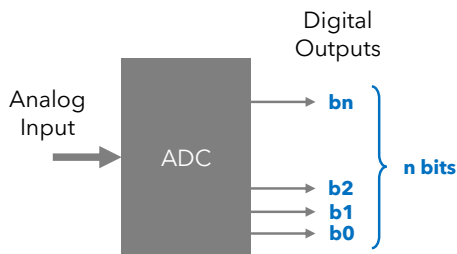
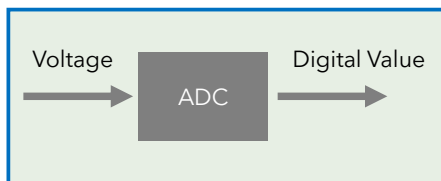
The **number of different values** that can be represented by **n bits** is **2^n** .

Example for $n = 3$ bits

Quantification



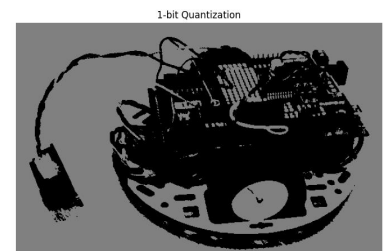
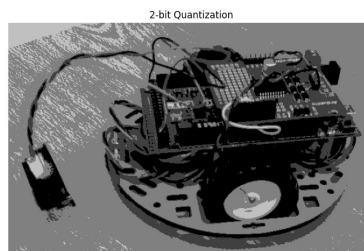
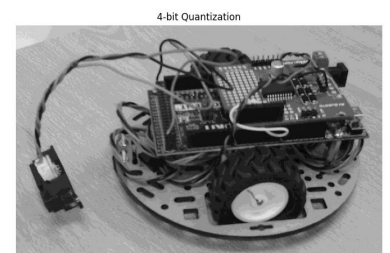
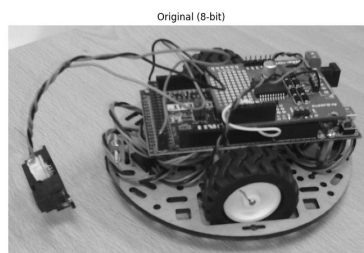
How an Analog to Digital Converter works ?

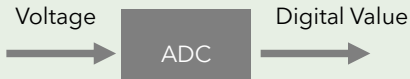


Each bit can have one of two values: **0** or **1**.

The **number of different values** that can be represented by **n bits** is **2^n** .

Quantification

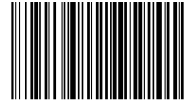




Sampling

Barcode to decode

Area of sampling

<https://barcode-coder.com/fr/specification-ean-13-102.html>


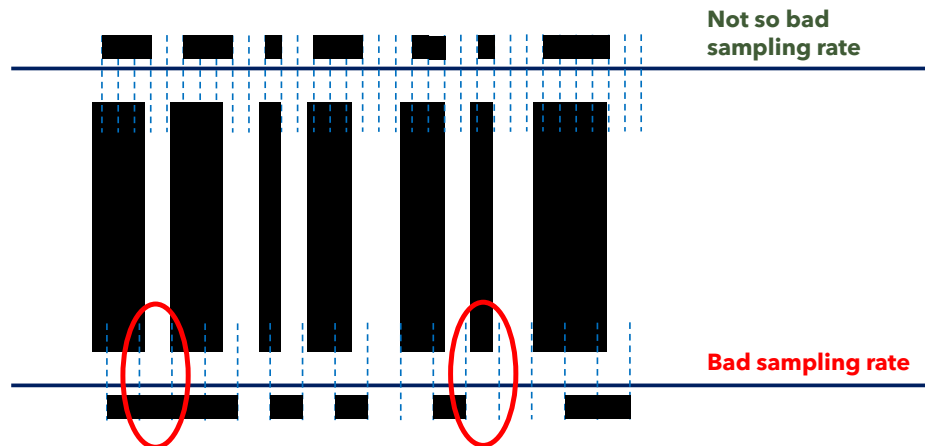
LEnSE 2024

Sampling theorem

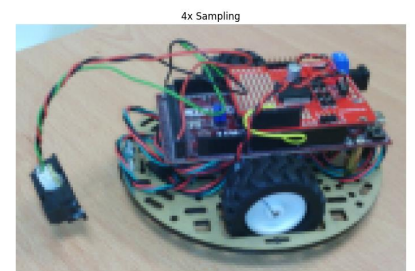
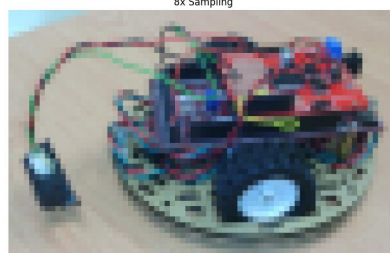
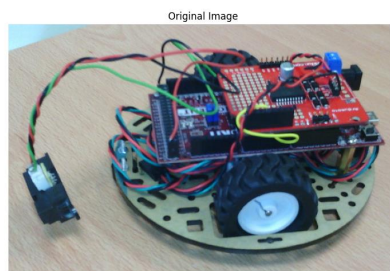
Nyquist-Shannon sampling theorem

The sampling frequency must be equal to or **greater than twice** the frequency associated with the finest detail in the image (edges).

With a grid spacing of d , a periodic component with a period higher than $2 \cdot d$ can be reconstructed.

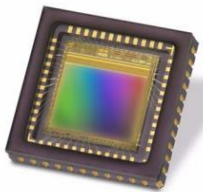
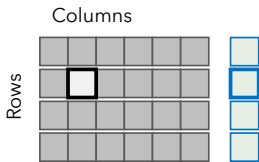


Sampling

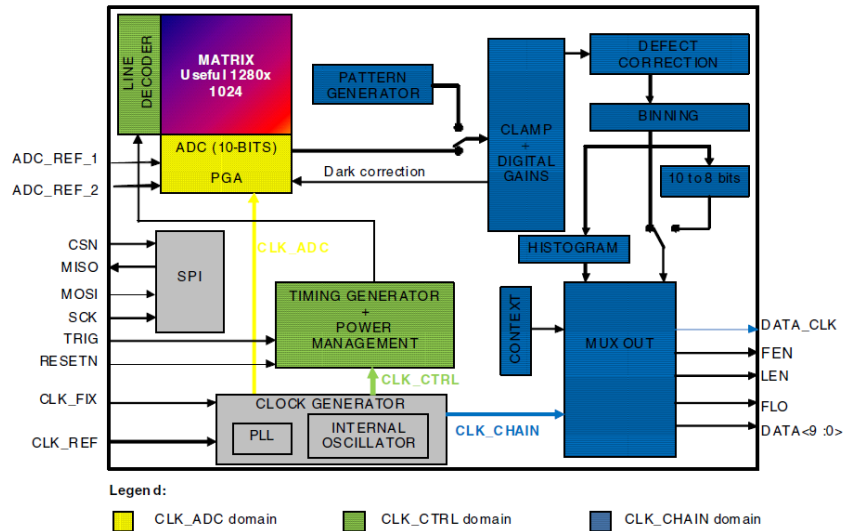


Cameras and Interfaces

Inside a real camera



e2v sensor EV76C560ACT



Cameras and Interfaces

Frame rate

Each image has a total amount of binary data :

$$\text{Nb of data (bits)} = \text{Nb of pixels} \times n$$

The amount of data per second :

$$\text{Nb of data per s (bits/s)} = \text{Nb of data (bits)} \times \text{FPS}$$

Example for a 4k camera in 12 bits @ 30 fps :

$$\text{Nb of data (bits)} = 3840 \times 2160 \times 12 = 99\,532\,800 \text{ bits}$$

$$\text{Nb of data per s (bits/s)} = 99\,532\,800 \times 30 = 2,9 \text{ billions of bits / s} = 2,78 \text{ Gbit/s}$$

In 2024, the transfer rate of a home router (optical fiber) is theoretically 8 Gbit/s (Free telecom - France)

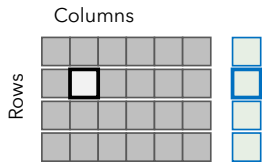
Frame rate

Number of individual frames captured per second by a device

Expressed in frames per second (fps)

Higher framerates result in smoother motion in video footage

Black level : an offset to compensate electronic defaults

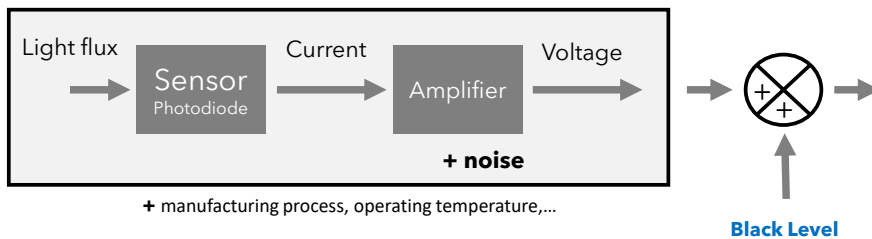


Dark Current

Response of the sensor to **complete darkness**

Black Level

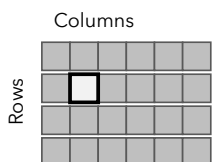
Change the **overall brightness** of an image.



Adjusting the camera's black level will result in **an offset to the pixel's gray values** output by the camera.

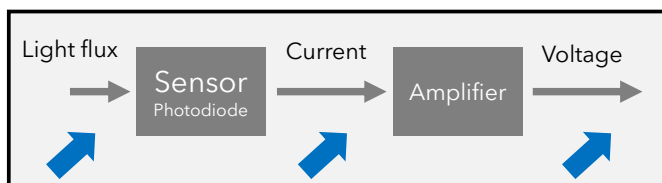
Due to **various physical and electronic factors**, the sensor's output is never zero, even in the complete absence of light

Exposure time and linearity

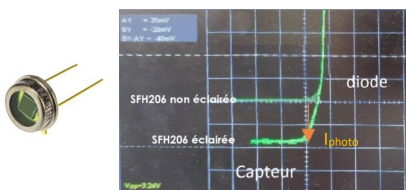


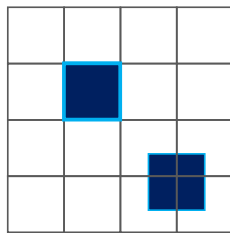
Exposure Time

Duration for which the **camera's sensor is exposed to light**, when capturing an image.



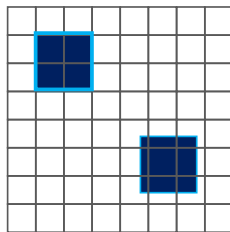
This parameter determines the amount of light collected.





Small object to detect

$$P = d$$



Security factor S

$$P = \frac{d}{S}$$

Spatial resolution / P

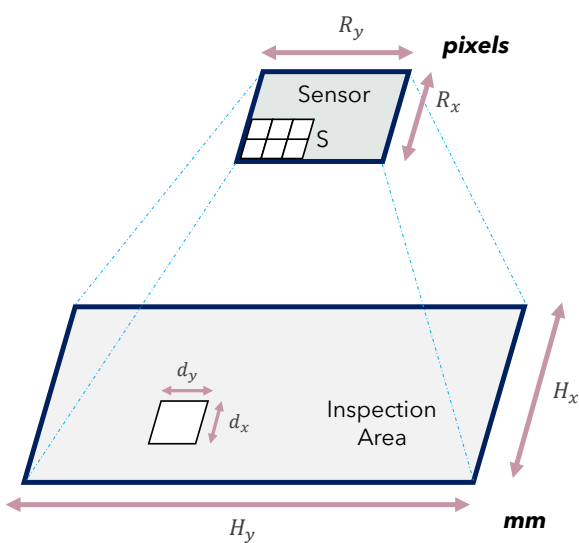
Distance observed by a single pixel in a given direction

This security factor is due to the Nyquist-Shanon theorem.

And $S \geq 2$



To verify if the spatial resolution is good enough, **calibration target** can be used. (Foucault)



Spatial resolution / P

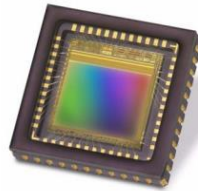
Distance observed by a single pixel in a given direction

$$P = \frac{d}{S}$$

Sensor resolution (pixels)

$$R = \frac{H}{P} = \frac{S \times H}{d}$$

H (mm) \rightarrow R (px)
 d (mm) \rightarrow S (px)
 P (mm) \rightarrow 1 (px)



e2v sensor EV76C560ACT

	Columns					
	1	2	3	4	5	6
1	Green	Blue	Green	Blue	Green	Blue
2	Red	Green	Red	Green	Red	Green
3	Green	Blue	Green	Blue	Green	Blue
4	Red	Green	Red	Green	Red	Green

