Institut d'Optique Graduate School
Interfaçage Numérique

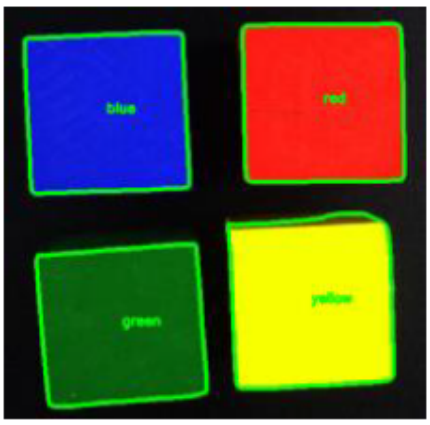
Interfaçage Numérique

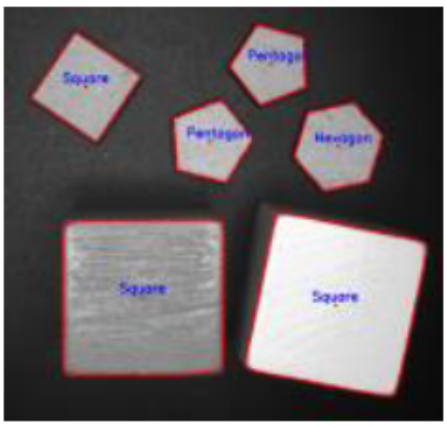
Travaux Pratiques

Semestre 6

Vision Industrielle

TP 2 / Bases de traitement d'images sous OpenCV





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 2 Vision Industrielle.

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/

Des codes en Python, proposant des exemples à tester, sont disponibles sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 2 Vision Industrielle / Répertoire vers codes à tester*.

Un fichier archivé, nommé STEP_BY_STEP.ZIP*, regroupe l'ensemble des codes à tester au cours de cette séance, ainsi que les images à traiter.*

Un **kit d'images** est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Bloc 2 Vision Industrielle / Kit d'images*.

Accumulation de preuves / Méthode de travail

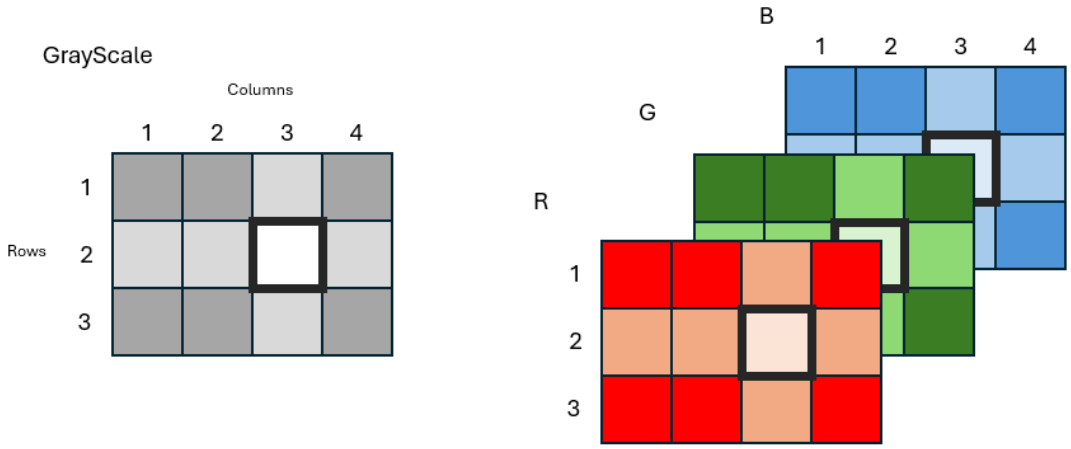
Il est conseillé pour ce TP de **créer un nouveau projet PyCharm** sur votre session (*attention à l'endroit où vous stockerez ce projet - U : sur les sessions Windows de l'IOGS*).

Les différents algorithmes proposés dans ce TP, ainsi que ceux que vous serez amenés à modifier ou créer, pourront vous resservir dans d'autres projets. Nous vous conseillons donc fortement de les **sauvegarder** précieusement et de les **commenter** autant que possible afin de retrouver rapidement les principes mis en jeu derrière les fonctionnalités de OpenCV.

Il serait également pertinent de votre part de rédiger un **journal de bord** sur ce TP en incluant les résultats (images, histogrammes...) et vos analyses des fonctionnalités et de leur intérêt en traitement d'images.

Rappel sur les images numériques

Une image RVB contient 3 canaux (Rouge, Vert, Bleu ou *RGB* en anglais), tandis qu'une image en niveaux de gris n'en a qu'un. Une image en niveau de gris sera **3 fois plus rapide** à analyser qu'une image en couleur RVB mais toute notion de couleur sera alors perdue.



La couleur des objets peut s'avérer inutile lorsqu'on cherche, par exemple, à détecter des formes particulières ou des contours dans une image.

De nombreux algorithmes d'analyse d'image ou de vision par ordinateur travaillent plus efficacement sur des images en niveaux de gris, permettant notamment d'uniformiser l'entrée des algorithmes et de réduire les informations redondantes liées à la couleur.

A - Ouvrir une image [20 min]

Notions : *Open an image* - *Display an image*

➜ **M** Ouvrir le fichier 01_OPEN_IMAGE.PY du répertoire des codes à tester.

➜ **M** Exécuter ce code.

➜ **Q** Que fait ce programme ? Quelle est la taille de chacune des images ? Quel est le type de données d'un élément ?

➜ **Q** Que vaut le premier pixel de chacune des images ? A quoi correspondent les données fournies ?

B - Changement d'espace de couleur [20 min]

➜ **M** Ouvrir et afficher l'image *couleurs_4.png* du kit d'images fourni, au format RGB.

➜ **M** Créer une copie de la matrice image (fonction *copy()* de Numpy).

➜ **M** Forcer à 0 tous les pixels du canal bleu de la copie de l'image et afficher la nouvelle image.

Plusieurs méthodes de conversion

Plusieurs méthodes existent pour passer d'une image RVB à une image en niveau de gris :
— Calculer la **moyenne des valeurs** des trois canaux de couleur (Rouge, Vert, Bleu) pour chaque pixel.
— Utiliser des **poids spécifiques pour les canaux R, V et B**, basés sur leur contribution relative à la perception humaine.
— Convertir l'image dans un **autre espace de couleur**, comme YUV, HSL ou HSV, et extraire la composante de luminosité.

Moyenne des canaux R,V,B

Cette méthode est la plus simple. Chaque pixel de l'image en gris est la moyenne des pixels des canaux rouge, vert et bleu de l'image en couleur :

$$Pixel_{Gray} = \frac{Pixel_R + Pixel_V + Pixel_B}{3}$$

➜ **M** Créer une image en nuance de gris utilisant la méthode de la moyenne des trois canaux.

➜ **M** Afficher l'image résultante.

Pondération en fonction de la perception humaine

Cette méthode est une moyenne pondérée des valeurs des pixels R, V, B de l'image couleur :

$$Pixel_{Gray} = 0.299 \cdot Pixel_R + 0.587 \cdot Pixel_V + 0.114 \cdot Pixel_B$$

Les coefficients de cette méthode proviennent de la sensibilité relative de l'œil humain aux différentes couleurs et ont été standardisés à l'origine pour la télévision analogique (NTSC). Ils sont aujourd'hui utilisés comme une approximation fidèle de la perception visuelle de la luminosité.

La méthode de conversion fournie par la bibliothèque OpenCV se base sur cette pondération.

➜ **M** Convertir l'image RVB en niveau de gris par l'instruction suivante :

```
1 image_gray = cv2.cvtColor(image_rgb, cv2.COLOR_BGR2GRAY)
```

➜ **M** Comparer les images obtenues par la moyenne classique et cette moyenne pondérée.

Utilisation d'un espace colorimétrique différent

L'espace colorimétrique RVB est très utilisé dans le domaine du numérique (affichage, acquisition d'images) pour sa facilité de mise en oeuvre.

Cependant, ce n'est **pas** le plus **adapté vis-à-vis de la perception humaine** où la luminance et la couleur sont séparées.

Des espaces comme YUV, YIQ, ou YCbCr séparent la composante de luminance (Y) des composantes de chrominance (U et V).

➜ **M** Convertir l'image RVB dans l'espace YUV par l'instruction suivante :

```
1  image_yuv = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2YUV)
```

➜ **M** Comparer alors l'image en niveau de gris obtenue par la méthode de moyennage pondérée et le canal Y de cette conversion.

➜ **Q** Que pouvez-vous conclure sur la méthode de calcul utiliser pour la luminance (Y) ?

Il existe d'autres espaces colorimétriques dans le domaine numérique. Voici un résumé non exhaustif :

Espace colorimétrique	Avantages
RGB	Simple, utilisé pour les écrans et le rendu des couleurs.
HSV / HSL	Intuitif pour manipuler la couleur (teinte, saturation).
YUV / YCbCr	Sépare luminance et chrominance.
CIE-Lab	Uniformité perceptuelle, idéal pour mesurer les différences de couleur.
CMY(K)	Optimisé pour l'impression.
XYZ	Modèle basé sur la perception humaine.

C - Calculer l'histogramme d'une image et l'afficher [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.*

Interfaçage Numérique

Travaux Pratiques

Semestre 6

Ressources

Bloc Vision Industrielle

Liste des ressources

— Camera BASLER a2A1920-uc/umBAS
— Source EFFI-Ring - Spectre et données / Version RGB
— Cubes de couleur - Réflectance

Camera BASLER a2A1920-uc/umBAS

La documentation complète se trouve sur le site du fabriquant - Basler

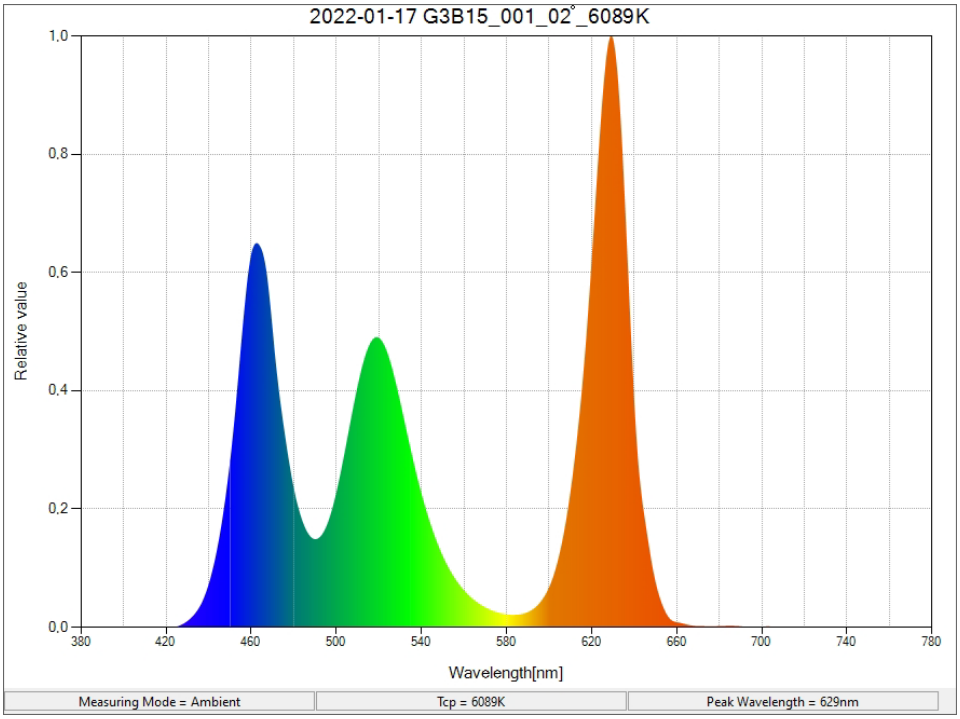
Marque Modèle	Basler a2A1920-160ucBAS
Résolution Taille pixel Profondeur	1920 px x 1200 px 3.45 x 3.45 μm^2 12 bits
Efficacité quantique Gain (1/K) Capacité de saturation Capacité de saturation	62.22 % 2.652 e-/DN 10492 e- 16862 p

Source EFFI-Ring - Spectre et données / Version RGB

La documentation complète se trouve sur le site du fabriquant - Effilux

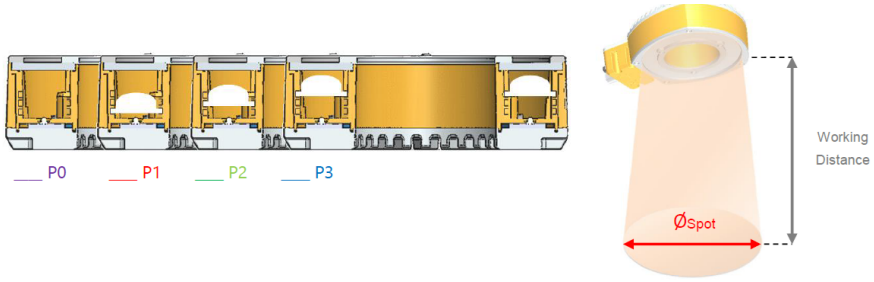
Spectre

Obtenu à l'aide d'un spectromètre



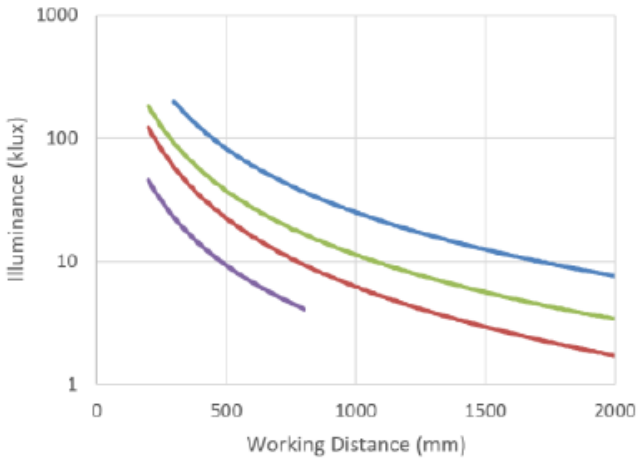
Taille du spot et éclairement en fonction de la distance de travail

Données provenant de la documentation technique.



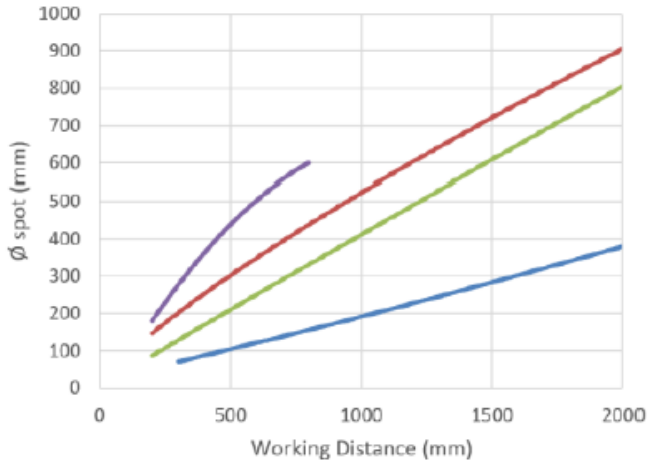
Illumination[1] vs. Working distance

Semi-Diffuse



$Ø_{spot}$[3] vs. Working distance

Semi-Diffuse



Cubes de couleur - Réflectance