

## Premières applications de vision sous OpenCV

### Partie 1

#### Exercice 1 : Chargement et affichage d'une image

Écrire une fonction qui permet de réaliser les opérations suivantes afin de charger et d'afficher une image :

1. Charger l'image **landscape.jpg**
2. Vérifier si l'image chargée est vide.
3. Créer une fenêtre graphique.
4. Afficher l'image dans la fenêtre graphique.
5. Fermer la fenêtre lorsqu'une touche clavier est appuyée.

#### Exercice 2 : Filtrage linéaire

L'objectif de cet exercice est de faire un filtrage linéaire d'une image bruitée. Écrire les instructions permettant de réaliser les traitements qui suivent :

1. Déclarer une image source, une image destination et une matrice noyau.
2. Initialiser le noyau, de taille 3x3, avec la valeur 1/9
3. Charger l'image **plane.jpg**
4. Vérifier si l'image n'est pas vide.
5. Appliquer un filtre linéaire 2D en utilisant le noyau prédéfini.
6. Afficher les images source et destination.

#### Exercice 3 : Lissage de l'image

Le lissage de l'image consiste à appliquer un filtre linéaire numérique pour atténuer le bruit dans l'image. Plusieurs filtres peuvent être appliqués, notamment, le filtre gaussien, médian et bilatéral.

1. Charger et vérifier l'image **monalisa.jpg**
2. Appliquer sur l'image source un filtre gaussien dont la taille du noyau est 5.
3. Appliquer sur l'image source un filtre médian dont la taille du noyau est 5.
4. Appliquer sur l'image source un filtre bilatéral, le diamètre du voisinage du pixel est 15, le paramètre de l'espace couleur est 15x2 et le paramètre de l'espace des coordonnées est 15/2.
5. Afficher les trois images résultats après l'application des trois filtres.

#### Exercice 4 : Morphologie mathématique

La morphologie mathématique est un ensemble de théorie et technique qui permet de transformer l'image, d'en extraire des caractéristiques et de mesurer, par une analyse associant les propriétés des objets eux-mêmes et les propriétés du contexte. Écrire une fonction qui permet de :

1. Charger et vérifier l'image **apple.png**
2. Définir un élément structurant carré de taille 7.
3. Dilater l'image source en utilisant l'élément structurant prédéfini en question 2.
4. Éroder l'image source avec le même élément structurant.
5. Afficher les images : source, dilatée et érodée.

### Exercice 5 : Seuillage de l'image

Le seuillage est une opération de segmentation simple qui transforme une image de niveaux de gris en image monochrome. Effectuer les opérations suivantes pour faire un seuillage de l'image :

1. Charger et vérifier l'image **tiger.jpg**
2. Convertir l'image source en niveaux de gris.
3. Appliquer le seuillage en utilisant le seuil 100 et un seuil maximal 255. Le type du seuillage est binaire.
4. Afficher l'image source et l'image résultat.

### Exercice 6 : Détection de contours

La détection de contour se définit par le calcul des points d'une image numérique qui correspondent à un changement brutal de l'intensité lumineuse. Écrire le code de la fonction de détection de contour en incluant toutes les opérations suivantes :

1. Charger et vérifier l'image **building.jpg**
2. Convertir l'image source en niveaux de gris.
3. Appliquer un filtre laplacien à l'image de niveaux de gris. La taille du noyau est 3 et la profondeur de l'image destination est 16S
4. Appeler la fonction qui permet la mise à l'échelle, le calcul de valeurs absolues et la conversion de l'image résultat en 8-bit.
5. Appliquer le filtre de Canny en utilisant les seuils 20 et 110.
6. Afficher les images source et résultats.

### Exercice 7 : Égalisation d'histogrammes

L'égalisation d'histogramme améliore et ajuste le contraste de l'image numérique. Écrire une fonction qui permet de :

1. Charger et vérifier l'image **bird.jpg**
2. Convertir l'image en niveaux de gris.
3. Appeler la fonction d'égalisation d'histogramme prédéfinie dans la librairie.
4. Afficher les images source et résultat.

## Partie 2

### Exercice 8 : Appariement d'images

Ce traitement consiste à trouver la correspondance entre une imagette, représentant une partie de l'image, et son image source. Le calcul d'appariement se base sur la similarité des fonctions d'intensités liées aux deux images. Réaliser une fonction qui permet d'effectuer l'appariement en se basant sur les traitements suivants :

1. Charger et vérifier l'image **bus.jpg** et son modèle **bus\_template.png**
2. Appeler la fonction qui permet l'appariement d'un modèle avec son image en utilisant la méthode des moindres carrées normalisées.
3. Appeler la fonction qui permet de trouver les coordonnées du modèle dans l'image destination.
4. Afficher un rectangle autour de l'emplacement du modèle dans l'image source.
5. Afficher les images source, destination et modèle.

### Exercice 9 : Détection de contours

L'objectif de cet exercice est d'appliquer un filtre de détection de contours sur l'image afin de segmenter les formes. Appliquer les traitements suivants pour réaliser cette segmentation par contours :

1. Charger et vérifier l'image **porsche.jpg**
2. Convertir l'image en niveaux de gris.
3. Appliquer le filtre de Canny en utilisant les seuils 100 et 200.
4. Appeler la fonction qui trouve les contours segmentés par le filtre de Canny.
5. Tracer les contours trouvés.
6. Afficher les images source et destination.

### Exercice 10 : Enveloppe convexe

Dans cet exercice, il est demandé de créer un nuage de points et de calculer son enveloppe convexe. Écrire une fonction qui réalise les opérations suivantes :

1. Déclarer une image de taille 500x500.
2. Dans une boucle infinie, créer un vecteur de points où les abscisses et les ordonnées sont des nombres aléatoires dont les limites inférieure et supérieure sont respectivement, 1/4 et 3/4 la largeur de l'image. Par ailleurs, les ordonnées sont des nombres aléatoires dont les limites inférieure et supérieure sont respectivement, 1/4 et 3/4 la hauteur de l'image.
3. Calculer l'enveloppe convexe de l'ensemble des points créés.
4. Afficher tous les points aléatoires générés.
5. Tracer l'enveloppe convexe.
6. Afficher l'image résultat.

### Exercice 11 : Appariement des descripteurs

Cette fonction consiste à appairer une image requête avec une image référence en utilisant les descripteurs locaux invariants. Écrire une fonction qui permet de réaliser les opérations suivantes :

1. Charger et vérifier les images : **box.png** et **box\_in\_scene.png**
2. Déclarer un détecteur de points d'intérêt de type ORB. Détecter les points d'intérêt des deux images.
3. Calculer les descripteurs des points d'intérêt des deux images.
4. Déclarer la fonction d'appariement de type Brute Force de norme L2. Calculer les appariements entre les deux vecteurs descripteurs.
5. Tracer les appariements et afficher l'image résultat.

**Exercice 12 : Classification de points avec les SVM**

Écrire les instructions permettant de réaliser la classification de points en utilisant le classifieur SVM.

1. Déclarer une image 512x512 et l'initialiser à 0.
2. Déclarer un vecteur d'étiquettes de 4 éléments {1, 2, 3, 4}.
3. Déclarer une matrice d'apprentissage dont les éléments sont : { {501, 10}, {255, 10}, {501, 255}, {10, 501} }.
4. Paramétrier un classificateur SVM avec, type : C\_SVC, noyau linéaire, critère de terminaison : 100 itérations.
5. Appeler la fonction d'apprentissage du classificateur.
6. Dans une boucle parcourant tous les pixels de l'image, calculer les réponses du classificateur en appelant la fonction de prédiction du classificateur. Selon les 4 réponses de la prédiction, affecter une couleur au pixel.
7. Afficher les points d'apprentissage et l'image résultat.

**Exercice 13 : Lecture vidéo et interface graphique**

Écrire une fonction qui permet de lire et d'afficher un flux vidéo. Réaliser les opérations suivantes pour effectuer le traitement :

1. Ouvrir et vérifier le flux **video.mp4**
2. Créer une fenêtre graphique et récupérer le nombre d'images du flux vidéo.
3. Créer un curseur de progression (slider).
4. Dans une boucle infinie, récupérer et vérifier les images de la capture.
5. Afficher l'image de la capture avec une temporisation de 40 ms.
6. Appeler la fonction qui met à jour le curseur de progression.
7. Définir la fonction callback liée au curseur. Cette fonction utilise la position du curseur pour positionner le flux sur l'image pointée.

**Exercice 14 : Fonction principale et menu du programme**

Ajouter une fonction de menu qui permet d'afficher l'ensemble des fonctions développées dans tous les exercices précédents. Dans la fonction principale du programme, écrire les instructions qui permettent d'appeler le menu et de sélectionner la fonction de traitement appelée selon le choix de l'utilisateur. Le menu est affiché en boucle tant que l'utilisateur souhaite continuer l'exécution du programme.