université PARIS-SACLAY
IUT DE CACHAN

Département
GEII 2

BUT3
Semestre 5

BUT Génie Électrique et Informatique Industrielle – ESE

Électronique

TP n°1 : FIR 1D et 2D

Auteur(s) : PR

Date : 2024
Version : V0.3

Prise en main du nano ordinateur Jetson Nano C100 :

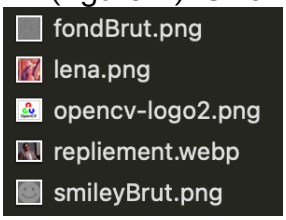
AVERTISSEMENT PRELIMINAIRE : les nano ordinateurs ont des systèmes d'exploitation facilement corruptibles. **Ne jamais DEBRANCHER l'alimentation du Jetson tant que sa led verte est allumée**, il faut toujours attendre son extinction complète même après avoir demandé l'arrêt à linux et après l'extinction du moniteur.

Procédure de démarrage :

1. Placer le Jetson à proximité de l'arrière du PC et libérer une prise secteur à proximité.
2. Débrancher le moniteur **côté PC** (laisser brancher le côté moniteur) et le brancher sur le DisplayPort du Jetson (ou sur son HDMI si c'est un câble HDMI).
3. Débrancher la souris et le clavier du PC et les brancher au Jetson.
4. S'assurer qu'aucun câble n'est tendu, ce qui peut casse la connectique.
5. Brancher l'alimentation pour Raspberry au secteur (ne pas utiliser un autre type d'alimentation).
6. Arrivé à l'écran d'accueil, rentrer le mot de passe : iut_GAUSS***XX (numéro marqué sur le jetson)***.

Premier programme :

1. S'assure grâce au navigateur que le répertoire `Images` contient les images pour le TP (figure 1). Sinon demander à l'encadrant.



2. Ouvrir la console.
3. Saisir la commande `gedit test.py`
4. Dans la fenêtre saisir le code ci—dessous :

```
import cv2
img = cv2.imread('/home/gaussXX/Images/lena.png', cv2.IMREAD_GRAYSCALE )
cv2.imshow('image',img)
cv2.waitKey(0)  #cette commande est imperative pour voir l image
                #0 : attend l'appui sur une touche
```

```
                #10 : attend 10ms
cv2.destroyAllWindows()
```

5. Enregistrer le fichier sur le bureau (commande ***save*** de gedit)
6. Revenir à la console, se placer dans le répertoire du bureau par la commande `cd Desktop` (**voir l'annexe B** pour en savoir plus sur les commandes Linux)
7. Une fois dans ce répertoire saisir : `python3 test.py`
8. L'image doit apparaitre à l'écran.

(pour la fin du TP) Procédure d'arrêt :

1. Utiliser la commande d'arrêt de Linux pour arrêter le Jetson.
2. **Attendre l'extinction complète de la led verte.**
3. Débrancher l'alimentation de la Jetson et de la prise secteur pour la ranger.
4. Débrancher le moniteur et le rebrancher au PC.
5. Débrancher la souris et le clavier et les rebrancher au PC.
6. Ranger le Jetson dans le carton.

Si vous voulez mettre en place OpenCV sur votre ordinateur :

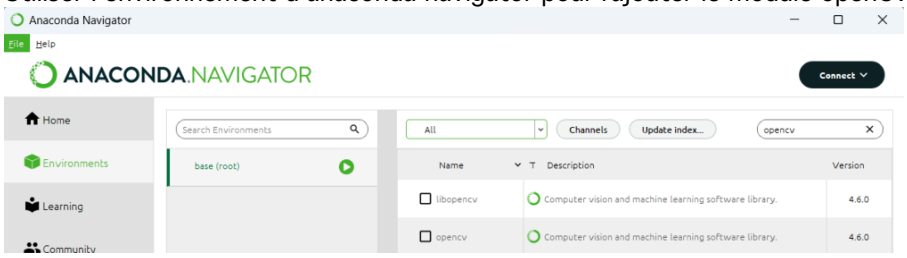
- Lancer Anaconda3 navigator.
- Dans anaconda trouver Spyder et cliquer sur LAUNCH.
- Dans la fenêtre « Console 1/A » de Spyder saisir la commande*.
  `pip install opencv-contrib-python`
- Lancer la commande suivante à partir du menu de Spyder :
  `Consoles/Restart kernell`
- Dans le fichier « temp.py » saisir le code :

```
import cv2
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE )
                              #choisir le chemin d une image

cv2.imshow('image',img)
cv2.waitKey(0)      #cette commande est imperative pour voir l image
                    #0 : attend l'appui sur une touche
                    #10 : attend 10ms
cv2.destroyAllWindows()
```

Autres possibilités :

Utiliser l'environnement d'anaconda navigator pour rajouter le module openCV



OU

1. Trouver CMD.exe dans anaconda navigator et cliquer sur LAUNCH
2. Lancer la commande suivante :
   `conda install -c conda-forge opencv`

PARTIE A - Étude de deux filtres élémentaires 1D puis 2D pour le traitement de l'image

1. *Premier objectif : En utilisant des boucles `for` imbriquées, écrire l'algorithme de floutage d'ordre 2 d'une image *ligne par ligne.*

Ci-dessous les éléments nécessaires pour atteindre l'objectif :

- Écrire ci-dessous la formule mathématique permettant de calculer le niveau de gris d'un pixel M[j,k] (notation Python) à partir des pixels de l'image notés E[j,k] pour un floutage d'ordre 2.

- Rappels de python :

  Attention à l'*overflow* (dépassement) : la somme d'uint8 conduit généralement à un dépassement de 255, ce qui provoque du *wrapping* sur les valeurs. Sur l'image c'est un effet de *moiré*.

  La boucle for

```
for x in range(2, 6):     # x va de 2 à 5
    …                     # le corps de la boucle est indenté
```

  Les tableaux 2D avec la bibliothèque numpy (np.array)

  Importer la bibliothèque : `import numpy as np`

  Déclaration :
```
arr = np.array([[1, 2, 3], [4, 5, 6]])
arr = np.zeros((3,3),dtype = np.uint8)#matrice zéro
#entiers non signés 8bits adaptés au niveaux de gris
```

  Un élément : `print('arr(j=1,k=2): ', arr[0, 1])`

  Dimension d'un array :
```
print (arr.shape)#tuple(nb lignes,nb col)
print (arr.shape[0])#nb lignes
```

2. *Deuxième objectif : En utilisant *une* boucle `for` imbriquée *supplémentaire*, écrire l'algorithme de floutage d'ordre 30 d'une image *ligne par ligne.*

3. *Troisième objectif : En utilisant *une* boucle `for` imbriquée *supplémentaire*, écrire l'algorithme de floutage d'ordre 1 en 2D.

Ci-dessous un élément nécessaire pour atteindre l'objectif :

- Écrire ci-dessous la formule mathématique permettant de calculer le niveau de gris d'un pixel noté M[j,k] à partir des pixels de l'image d'entrée de pixels notés E[j,k] (voir TD1 question 9).

* Les questions avec * sont au programme du DS pour tous les apprentis. Les questions avec ** uniquement pour les BUT3.

4. **Quatrième objectif : à partir du programme précédent appliquer un filtre **d'ordre 3** en 2D en utilisant **la représentation matricielle** d'un filtre.

Ci-dessous les éléments nécessaires pour atteindre l'objectif :

- Écrire ci-dessous sous forme de matrice le filtre moyenneur 2D du 1er ordre noté F (voir TD1 question 10).

- Rappels de python pour créer la matrice F:

  Les tableaux 2D

  Une matrice de uns:

  ```
  F = np.ones((7,7),dtype = np.uint8)
  ```

  Une matrice de 1/9:

  ```
  F = np.ones((7,7),dtype = np.uint8)/9
  ```

5. *Cinquième objectif : Utiliser la fonction de *convolution* matricielle de openCV pour réaliser le même floutage.

Ci-dessous les éléments nécessaires pour atteindre l'objectif :

- Écrire ci-dessous l'expression mathématique correspondant à la convolution entre une matrice E et une matrice F, le résultat en M.

- **Syntaxe de convolution de openCV* :**

  ```
  M = cv2.filter2D(E, -1, F)
  ```

6. **Sixième objectif : Utiliser la *fonction de floutage (ang. blur)* intégrée à openCV.
   - **Syntaxe openCV :**

     ```
     M = cv2.blur(E,(3,3))
     ```

7. **Septième objectif : Utiliser la fonction de *convolution* matricielle de openCV pour réaliser la détection de contour 2D d'ordre 1.

   - La formule mathématique est la suivante :

$$D[j][k] = \frac{|-E[j-1][k] - E[j][k-1] + 4 * E[j][k] - E[j+1][k] - E[j][k+1]|}{4}$$

* The function does actually compute correlation, not the convolution. If you need a real convolution, flip the kernel using flip and set the new anchor to (kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1).

- Écrire sous forme de matrice le filtre dérivateur 2D du 1er ordre noté F (voir TD1):

8. ***Huitième objectif :** Utiliser les fonctions de détection de contour intégrées à openCV en les appliquant aux **deux images suivantes** : lena.png puis opencv-logo.png.

   - **Syntaxe openCV :**

   ```
   D = cv2.laplacian(E, cv2.CV_64F)
   ```

   - **Syntaxe openCV avec en plus un code de seuillage :**

   ```
   D = cv2.canny(E, 50, 140)
   ```

   En consultant la documentation openCV écrire ci-dessous les rôles des paramètres 50 et 140.

9. **Neuvième objectif (si le temps le permet)** : En utilisant des boucles for imbriquées, écrire l'algorithme de détection de contour d'ordre 1 *ligne par ligne.*

   - La formule mathématique est celle du filtre « dérivateur » d'ordre 1 :
     D[k] = **|** E[k] – E[k-1] **|** (où **|**...**|** est la valeur absolue).

   - Rappels de python :
     Valeur absolue : `abs(…)`
     Conversion en float : `float(…)`
     Attention : il y a un risque important d'overflow car la soustraction donne du négatif mais les uint8 ne peuvent coder que du positif. Il faut donc convertir les valeurs uint8 en float avant le calcul.

ANNEXE A: COMPLEMENTS SUR LES MANIPULATIONS DE MATRICES

Conversion en uint8 : `np.uint8(…)`
Concaténer deux array : `np.concatenate((np.zero(2,2),arr),axis=1)`

```
                        #concaténation par les colonnes car axis1
                        #le nombre de lignes doit être identique
```

Copier une matrice A dans une matrice B :

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9] ])
B = np.zeros((5,5),dtype = np.uint8)
B[1:4,1:4]=A #se lit: les cases des lignes 1 à 3 de B sur
             #leurs colonnes 1 à 3 sont égales aux cases de A
```

ANNEXE B: Commandes de base Linux

`ls` liste le contenu de répertoires.
Syntaxe : `ls [o ptions] [répertoire1] [répertoire2]`
`ls a` liste toutes les entrées y compris les entrées cachées.
`ls l` liste les entrées et affiche toutes les informations.

La commande `cd` permet le positionnement sur un répertoire
syntaxe : `cd [répertoire]`
`cd /` positionnement à la racine
`cd ..` positionnement sur le répertoire parent
`cd /home/desktop` positionnement depuis la racine sur le répertoire `/home/desktop`
`cd ../lib` positionnement à partir du répertoire parent sur le répertoire `/lib`.
`cd cd~ ou cd $HOME` positionnement depuis n'importe où sur le répertoire personnel.
`cd -` positionnement sur le répertoire précédent

La commande `pwd` affiche le nom absolu du répertoire courant, elle vous permet de connaître votre position dans l'arborescence.
syntaxe : `pwd`

La commande `cat` permet de visualiser le contenu d'un fichier texte
Syntaxe : `cat fichier fichier1 [fichier2]`

La commande `file` permet d'obtenir la nature du ou des fichiers spécifiés.
syntaxe : `file fichier1 [fichier2] [fichierN]`

La commande `sudo` permet à tout utilisateur de passer en mode Super Utilisateur. Avec l'option ''- '' les fichiers profile sont pris en compte.