Activité : traitement des images ; NSI

Dans cette activité, nous allons utilier Python pour traiter des images.

I/ Apprendre les bases : afficher une image avec Python

On considère le programme ci-dessous (fichier **Programme Images 1.py**)

```
1  import matplotlib.image as mpimg # On charge la bibliothèque matplotlib.image
2  import matplotlib.pyplot as plt # On charge la bibliothèque matplotlib.pyplot
3  import numpy as np # On charge la bibliothèque numpy
4
5  image = mpimg.imread("Images\Paysage 1.png") # On lit le fichier Paysage 1.png situé dans le dossier Images
6
7  if image.dtype == np.float32: # Si le résultat n'est pas un tableau d'entiers
8      image = (image * 255).astype(np.uint8) # On effectue une conversion des données
9
10 tailleImage = image.size # Taille de l'image : données utiles en octets
11 nombreLignes = image.shape[0] # nombre de lignes
12 nombreColonnes = image.shape[1] # nombre de colonnes
13
14 print(tailleImage) # On affiche la taille de l'image
15 print(nombreLignes) # On affiche le nombre de lignes
16 print(nombreColonnes) # On affiche le nombre de colonnes
17
18 imageTraitee = np.copy(image) # On fait une copie de l'image originale
19
20 for i in range(imageTraitee.shape[0]): # On parcours les
21     for j in range(imageTraitee.shape[1]): # On parcours les
22         r, v, b, n = image[i, j] # On extrait les composantes R V B du pixel
23         imageTraitee[i, j]=(r,v,b,255) # On recopie le pixel
24
25 plt.imshow(imageTraitee) # On affiche l'image traitée
26 plt.show()
27
28 mpimg.imsave("Images\imageTraitee 1.png", imageTraitee) # On sauvegarde l'image traitée
```

a) Lire chacune des lignes de ce programme et comprendre son fonctionnement
b) - Quelle instruction permet de lire une image ?
- Quelle instruction permet de copier une image ?
- Quelles instructions permettent de d'afficher une image ?
- Quelle instruction permet de sauvegarder une image ?
- Quelle instruction permet d'extraire les caractéristiques d'un pixel situé ligne i ; colonne j ?
- Quelles instructions permettent de connaître le nombre de lignes ; de colonnes et le nombre total d'octets correspondant à l'image ?

c) Coder ce programme en langage Python et testez-le.
d) Par quelles informations sont représentés chaque pixel de l'image ?
e) Que fait ce programme ?

II/ N'afficher qu'une composante de couleur d'une image avec Python

1°) Reprendre le programme ci-dessous et modifier la ligne 23 de manière à n'afficher que la composante rouge de l'image.

2°) Modifier de nouveau le programme ci-dessous de manière à n'afficher que la composante verte de l'image.

3°) Modifier encore une fois le programme ci-dessous de manière à n'afficher que la composante bleu de l'image.

On validera des différents programmes en réalisant des tests sur les différentes images se trouvant dans le dossier Images.

III/ Vision du chien

Les chiens ne voient pas les mêmes couleurs que nous. Pour simplifier, ils ne voient pas la composante rouge. Ecrire un programme en Langage Python qui affiche l'image telle que les chiens la perçoivent .

IV/ Afficher l'image en niveaux de gris

Nous allons effectuer la transformation suivante :

$(R\,;V\,;B)\rightarrow(I\,;I\,;I)$ où $I=\dfrac{\sqrt{R^2+V^2+B^2}}{\sqrt{3}}$

a) En reprenant les programmes développés ci-dessus, coder en langage Python un programme qui convertisse et affiche l'image en niveaux de gris.

b) Valider en testant le programme sur les différentes images se trouvant dans le dossier Images.

V/ Afficher l'image en noir ou en blanc

Nous allons effectuer la transformation suivante :

$(R\,;V\,;B)\rightarrow(c\,;c\,;c)$ avec $I=\dfrac{\sqrt{R^2+V^2+B^2}}{\sqrt{3}}$ et :

- si $I\geq$seuil alors $c=255$ et
- si $I<$seuil alors $c=0$

a) En reprenant les programmes développés ci-dessus, coder en langage Python un programme qui convertisse et affiche l'image en noir ou en blanc.

b) Valider en testant le programme sur les différentes images se trouvant dans le dossier Images.

VI/ Pour aller plus loin : détection d'un pixel

A partir de l'image dans le fichier **Trouver pixel blanc.png**, coder un programme en langage Python qui indique la position (numéro de ligne et numéro de colonne) du pixel blanc présent dans l'image.

VII / Création d'images

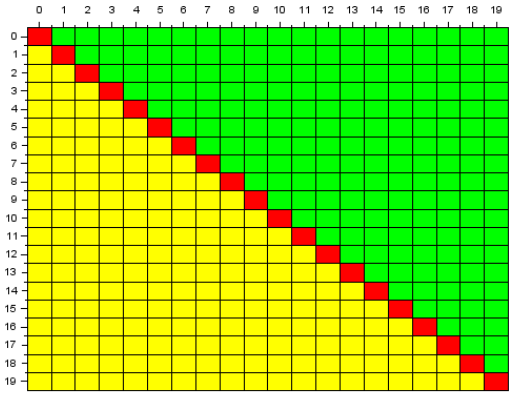
On va maintenant travailler avec une image de taille 20 x 20
On utilisera le fichier **Image 20 20.png** pour travailler sur l'image de 20 lignes ; 20 colonnes.

Les pixels sont numérotés de 0 à 19, est-ce cohérent ?

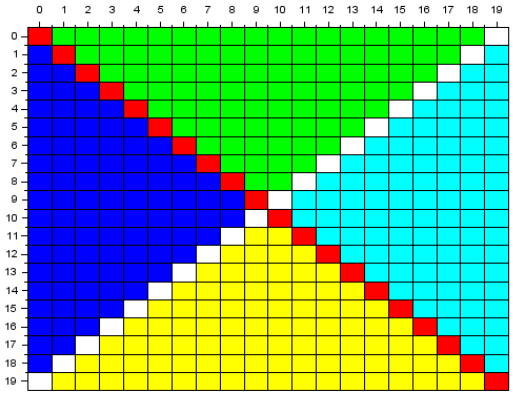
a) Écrire en langage Python un programme qui dessine une diagonale rouge qui part du pixel (0;0) au pixel (19;19)

b) Écrire en langage Python un programme qui dessine une diagonale verte qui part du pixel (0;19) au pixel (19;0)

c) Écrire en langage Python un programme qui dessine la figure ci-dessous
La diagonale est rouge
Le triangle inférieur gauche est jaune
Le triangle supérieur droit est vert



d) Écrire en langage Python un programme qui dessine la figure ci-dessous
La diagonale descendante est rouge et la diagonale ascendante est blanche
Le triangle de gauche est bleu ; le triangle de droite est cyan
Le triangle en haut est vert et le triangle en bas est jaune



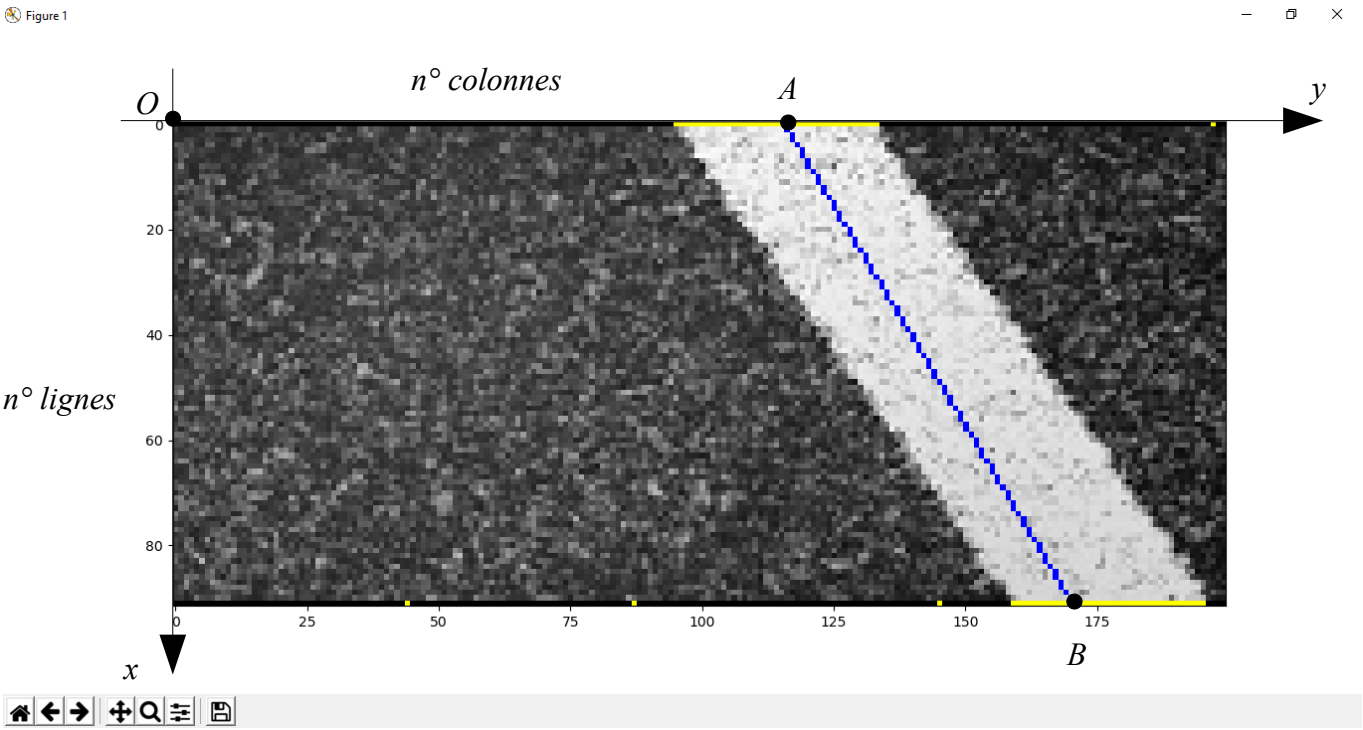
e) Écrire en langage Python un programme qui dessine un cercle défini par la position de son centre ; son rayon exprimés en pixels et sa couleur exprimée en coordonnées R ; V et B.

VIII/ Intelligence artificielle : aide au maintien dans la voie

Afin de respecter les nouvelles normes en terme de sécurité routière, les automobiles sont de plus en plus équipées de systèmes d'aides automatiques à la conduite comme par exemple le maintien dans la voie. Ce dispositif est censé empêcher un véhicule de sortir de la route : pour ceci, une caméra située derrière le rétroviseur intérieur filme les marquages au sols (bandes blanches) et à partir de ces informations effectue si nécessaire une correction de trajectoire en agissant sur la direction de la voiture.

A partir de l'image dans le fichier **Trouver ligne blanche.png**, coder un programme en langage Python qui donne l'équation réduite de la ligne blanche dans le repère de l'image.

Indices : trouver deux points notés A et B assez éloignés sur la ligne blanche, relever leurs coordonnées et déterminer l'expression algébrique de la fonction affine représentée par la droite (AB).



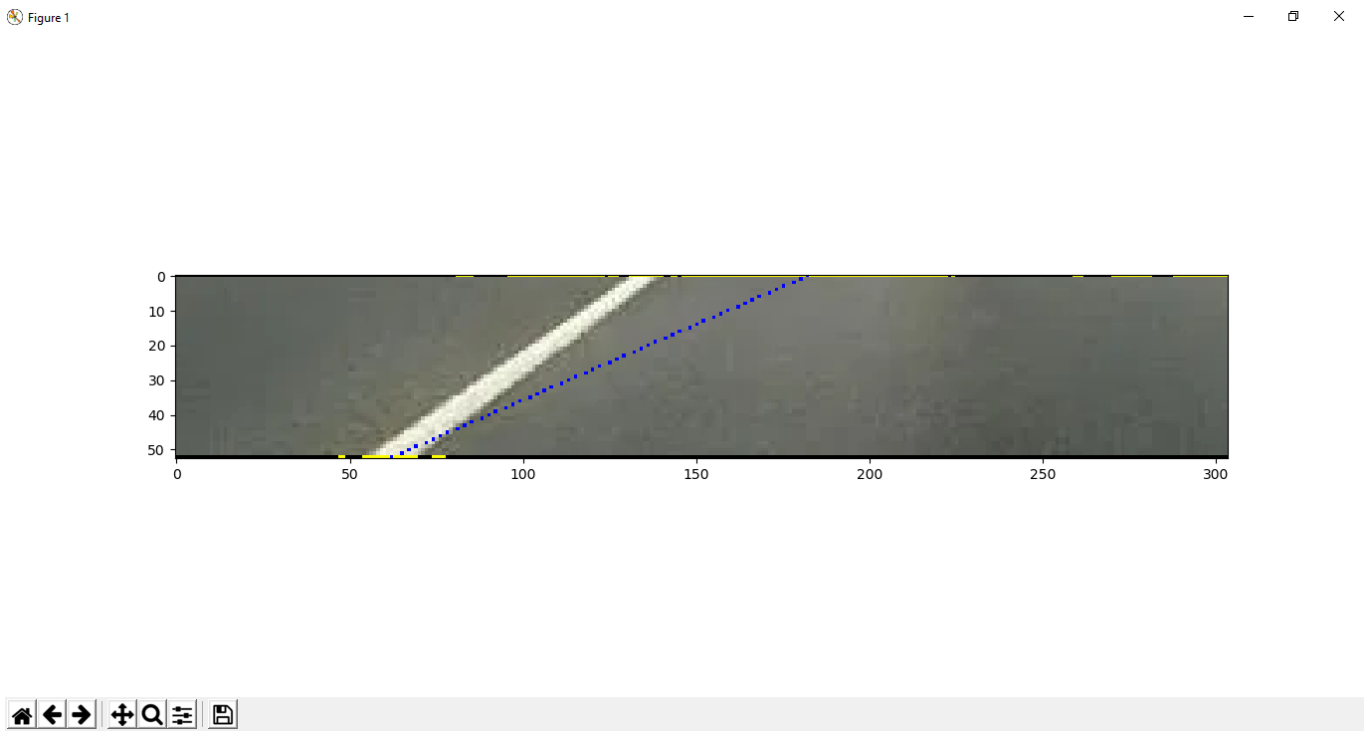
Ci-dessus : un ligne blanche rectiligne vue par l'orinateur (en bleu).

```
# Début du code : on charge des bibliothèques et on ouvre le fichier image

import matplotlib.image as mpimg # On charge la bibliothèque matplotlib.image
import matplotlib.pyplot as plt # On charge la bibliothèque matplotlib.pyplot
import numpy as np # On charge la bibliothèque numpy
image = mpimg.imread("Images\Trouver ligne blanche 1.png") # On lit le fichier Paysage 1.png situé dans le dossier Images
if image.dtype == np.float32: # Si le résultat n'est pas un tableau d'entiers
    image = (image * 255).astype(np.uint8) # On effectue une conversion des données
tailleImage = image.size # Taille de l'image : données utiles en octets
nombreLignes = image.shape[0] # nombre de lignes
nombreColonnes = image.shape[1] # nombre de colonnes
print(tailleImage) # On affiche la taille de l'image
print(nombreLignes) # On affiche le nombre de lignes
print(nombreColonnes) # On affiche le nombre de colonnes
imageTraitee = np.copy(image) # On fait une copie de l'image originale


# Un premier exemple de code pour trouver les coordonnées du point A

# On parcourt la première ligne
i = 0 # première ligne : ligne n°0
xA = 0 # position de la première ligne dans le repère de l'image
num = 0
den = 0
for j in range(imageTraitee.shape[1]): # On parcours les colonnes de la première ligne
    r, v, b, n = image[i, j] # On extrait les composantes R V B du pixel
    intensitePixel = ((r**2 + v**2 + b**2)**0.5)/(3**0.5)
    print(intensitePixel)
    if(intensitePixel > 110): # On a trouvé un pixel blanc (la ligne blanche)
        imageTraitee[i, j]=(255,255,0,255)
        num = num+j # On fait la somme des positions (n° de colonnes) des pixels blancs de la première ligne
        den = den+1 # nombre de pixels blancs sur la première ligne
print(num)
print(den)
yA = num/den # Position moyenne des pixels blancs sur la première ligne

# Un deuxième exemple de code pour trouver les coordonnées du point A
# A faire

# Un troisième exemple de code pour trouver l'équation de la droite (AB) : y = m x + p
# A faire


# Un quatrième exemple de code pour trouver tracer la ligne vue par l'ordinateur

# On trace la ligne blanche "vue" par le programme Python en bleu
# On ne traite pas le cas où la ligne blanche est verticale dans le repère de l'image
for i in range(imageTraitee.shape[0]): # On parcours les lignes : x
    x=i
    y = m*x+p
    j = int(y) # y peut être un nombre à virgule, on arrondit à l'entier le plus proche
    imageTraitee[i, j]=(0,0,255,255) # On trace un pixel bleu
plt.imshow(imageTraitee) # On affiche l'image traitée
plt.show()
```

<u>Problème de contraste :</u> sur l'image **Trouver ligne blanche 3.png**, le programme fonctionne mal car il y a des problèmes de contrastes entre la ligne blanche et la route : certains éléments de la route sont trop clairs.

Voici ce qu'on obtient avec un seuil fixe de 110 pour détecter le pixels de la ligne blanche : l'ordinateur croit que certains pixels de l'asphalte correspondent à la ligne blanche.

Ci-dessous : erreur d'analyse de l'ordinateur

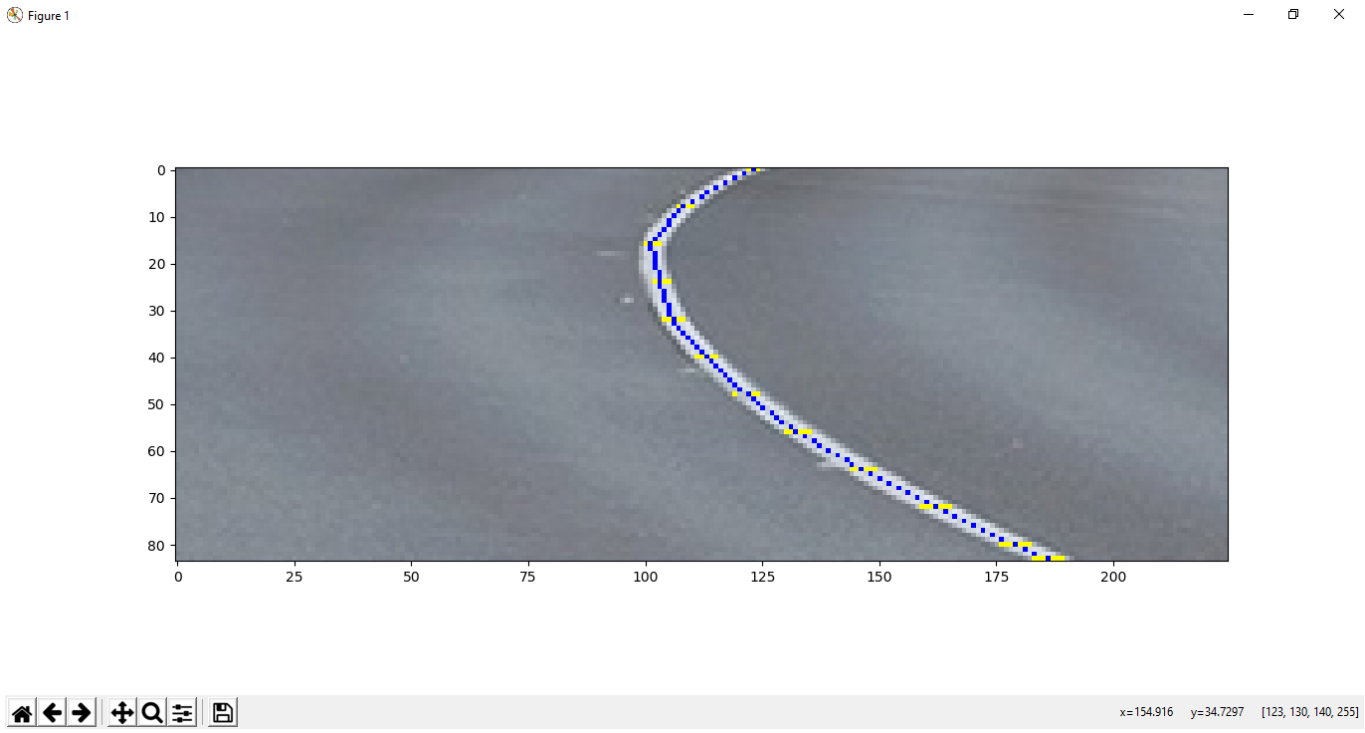


Pour rendre la détection de la ligne blanche plus robuste, on va considérer le programme ci-dessous

```
# On parcourt la première ligne
i = 0 # première ligne : ligne n°0
# On cherche le maximum
maximum = 0
for j in range(imageTraitee.shape[1]): # On parcours les colonnes de la première ligne
    r, v, b, n = image[i, j] # On extrait les composantes R V B du pixel
    intensitePixel = ((r**2 + v**2 + b**2)**0.5)/(3**0.5)
    if(intensitePixel > maximum):
        maximum = intensitePixel
seuil = 0.9*maximum

xA = 0 # position de la première ligne dans le repère de l'image
num = 0
den = 0
for j in range(imageTraitee.shape[1]): # On parcours les colonnes de la première ligne
    r, v, b, n = image[i, j] # On extrait les composantes R V B du pixel
    intensitePixel = ((r**2 + v**2 + b**2)**0.5)/(3**0.5)
    print(intensitePixel)
    if(intensitePixel > seuil): # On a trouvé un pixel blanc (la ligne blanche)
        imageTraitee[i, j]=(255,255,0,255)
        num = num+j # On fait la somme des positions (n° de colonnes) des pixels blancs de la première ligne
        den = den+1 # nombre de pixels blancs sur la première ligne
    else:
        imageTraitee[i, j]=(0,0,0,255)
print(num)
print(den)
yA = num/den # Position moyenne des pixels blancs sur la première ligne
```

Gestion des courbes :

Améliorer le programme de manière à ce qu'il découpe (« segmente ») l'image en tranches horizontales et trace plusieurs segments sur chacune des tranches horizontale.



Ci-dessus : un ligne blanche dans une courbe vue par l'orinateur (en bleu).