G4 : Jean-François Lance – Dominique Moinet - Luc Chevalot – Hassan El Fatihi - Juan-José Trives

Traitement d'images – Algorithmes

Fiche d'identité

Résumé de l'activité	Développer différents algorithme de traitement d'images.
Objectif(s)	1. Rappel du codage d'une image 2. Analyse d'algorithmes appliqués à des images 3. écrire les algorithmes permettant différentes transformations sur des images (symétries axiales, seuillage, Niveau de gris, contour)

Durée approximative de l'activité	• 2x2 heures
Participants	• Elèves de 1ère NSI - Un groupe de 24 au plus
Matériel nécessaire	• Un poste informatique • Editeur Python • Editeur/Visionneur graphique
Prérequis	• Codage Python, listes, notions mathématiques symétries

Notions liées (Contenus du programme)	P-uplets Tableau indexé Constructions élémentaires Parcours séquentiel d'un tableau
Lien avec le programme scolaire	Ecrire une fonction renvoyant un p-uplet de valeurs

(Capacité attendue)	Lire et modifier les éléments d'un tableau grâce à leurs index Utiliser des tableaux de tableaux pour représenter des matrices Itérer sur les éléments d'un tableau Mettre en évidence un corpus de constructions élémentaires Ecrire un algorithme de recherche d'une occurrence sur des valeurs de type quelconque

Objectifs : **A travers un premier exemple donné, faire découvrir quelques fonctions usuelles de traitement d'images.if(valeur < 128):**

Durée de la séquence : **3 heures**

Elèves concernés : **classe 1ère NSI**

Prérequis : **Langage python (les bases + listes) - Algorithmique**

Ressources logicielles :

Visual Studio Code (écriture code python)
Gimp (Editeur-Visionneur graphique)

INTRODUCTION

L'imagerie est un domaine de l'informatique qui concerne l'acquisition, l'analyse et le traitement des images en deux dimensions (2D). Le traitement d'images consiste à modifier certains pixels de pour améliorer la qualité de l'image (ébavurage). L'analyse d'images consiste à lire, à examiner les pixels d'une image sans les modifier (détection d'objets, extraction de contours).

Domaines d'application: Imagerie médicale, météorologie, géologie,....., robotique et reconnaissance de formes. Dans ce dernier cas, c'est le domaine de la **vision** (3D).

LE TP

Installation

Vérifier la présence des logiciels suivant sur votre ordinateur, sinon installez-les :

→ Visual Studio Code (VSCodeUserSetup-x64-1.35.1.exe)
→ Gimp (gimp-2.10.12-setup-1.exe)

Qu'est-ce qu'une image ?

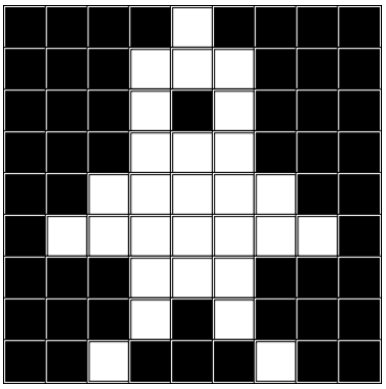
Une image est un ensemble de pixels, de dimension connue (ex 1600*1200)
Une image est soit :

- En noir et blanc : dans ce cas, les pixels constituant l'image sont codés sur 2 valeurs (0 pour blanc, 1 pour noir).
- En niveaux de gris : dans ce cas, les pixels constituant l'image sont codés sur 256 valeurs (de 0 à 255). On obtient un dégradé de différents gris.
- En couleur : Une image couleur est codée sur un triplet (r,v,b) correspondant à la composante rouge, vert, bleu. Les valeurs correspondent à l'intensité des couleurs.

Nous considérons ici qu'une image est codée à l'aide d'un tableau de points, on parlera de codage matriciel (à l'aide d'une matrice) ou bitmap.

L'image est décrite point par point. Il s'agit de la façon la plus simple de coder une image. Ces points sont appelés pixels. Chaque pixel est caractérisé par un nombre ou une liste de nombres indiquant sa couleur.

Ci-dessous, à gauche, une image en noir et blanc et à droite sa matrice.



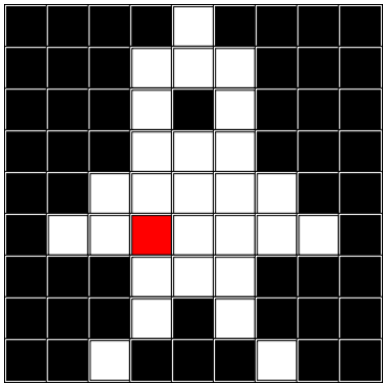
1	1	1	1	0	1	1	1	1
1	1	1	0	0	0	1	1	1
1	1	1	0	1	0	1	1	1
1	1	1	0	0	0	1	1	1
1	1	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	1	1
1	1	1	0	1	0	1	1	1
1	1	0	1	1	1	0	1	1

L'image est composée de 9 pixels en largeur et 9 pixels en hauteur.
Elle est codée par un tableau de 9 lignes et 9 colonnes.
Chaque pixel est accessible par ses coordonnées (i, j) dans le tableau :
- i étant la coordonnée horizontale ;
- j, la coordonnée verticale.

Attention l'indexation commence à 0 : le premier pixel en haut à gauche a pour coordonnées (0, 0) !

Exemple : Nous considérons ici qu'une image est codée à l'aide d'un tableau de points, on parlera de codage matriciel (à l'aide d'une matrice) ou bitmap.

Exemple :



Si l'image ci-contre est appelée *Img*.

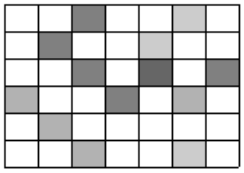
Pour accéder au pixel rouge, on écrira Img[5,3].

En effet, les coordonnées de ce pixel sont (5,3)

Dans les activités suivantes nous considérerons que les images utilisées sont représentées par des tableaux de nombres.

Activité débranchée :

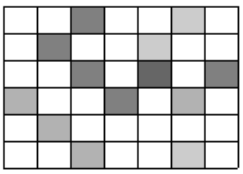
On considère une image nommée Img, constituée de 7 pixels en largeur, sur 6 pixels en hauteur.
L'image Img sera donc une liste de 6 listes de 7 pixels.
Img[i][j] désigne le pixel qui se situe sur la i ème ligne et j ème colonne (l'indexation commençant à 0)



On considère les algorithmes suivants :

Algo1

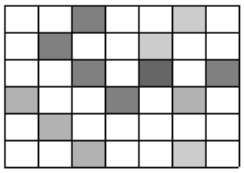
Pour i allant de 0 à $n-1$
  Pour j allant de 0 à $m-1$
    si i $i = 0$ ou $i = n-1$ ou $j = 0$ ou $m-1 = 0$
      le pixel $\text{Img}[i][j]$ devient noir



Question 1
Appliquer cet algorithme à l'image ci-contre

Algo2

Pour j allant de 0 à $m-1$
  les pixels $\text{Img}[0][j]$ et $\text{Img}[n-1][j]$ deviennent noirs
Pour i allant de 0 à $n-1$
  les pixels $\text{Img}[i][0]$ et $\text{Img}[i][m-1\ ]$ deviennent noirs



Question 2
Appliquer cet algorithme à l'image ci-contre

Question 3
Que pensez de ces deux exécutions ?

Question 4
On souhaite maintenant donner une épaisseur de k pixels au bord de notre image (sans en changer la taille). Ecrire un algorithme Algo3 permettant de réaliser cette opération.

Question 5
Pour aller plus loin :
Ecrire des algorithmes Algo5 permettant de tracer une ligne noire verticale au milieu de l'image et Algo6 permettant de tracer une ligne noire horizontale au milieu de l'image.
(On fera attention à la parité du nombre de pixels, la ligne devant partagée l'image en deux parties de même nombre de pixels)
Et pour les diagonales ?

TP programmation

Exercice 1 : Transformation d'une image de niveaux de gris vers une image en noir et blanc :

Comment transformer une image de niveaux de gris en noir et blanc ?
Il faut appliquer à cette image un traitement appelé seuillage.

Principe :
Tous les pixels d'une image en niveaux de gris ayant une valeur comprise entre 0 et 127 prendront pour valeur 0 lors de la transformation.
Tous les pixels d'une image en niveaux de gris ayant une valeur comprise entre 128 et 255 prendront pour valeur 1 lors de la transformation.

On peut schématiser cette opération de la façon suivante :
Exemple: 256 niveaux de gris en entrée → 2 niveaux (noir ou blanc) en sortie du traitement.

f(i)	→ g(i)
0	0
.	.
.	.
.	.
.	.
127	0
128	255
.	.
.	.
.	.
255	255

Dans ce cas on réalise un seuillage. Une table permet de faire une conversion de niveau.
On a transformé une image de niveaux de gris en noir et blanc
Application : Améliore le visuel de l'image
Avantage : l'image pourra être compressée de façon optimale.

Travail demandé :

➔ Ecrire un algorithme simple permettant de transformer de niveau de gris en un pixel noir et blanc.

/* En entrée : valPixelGris */
/* En sortie : valPixelNB */

Le code source de l'application est :

```
from PIL import Image

imageSource=Image .open("Dog.pgm")
largeur , hauteur=imageSource.size
print(largeur,hauteur)
imageNB=Image.new("L",(largeur,hauteur))

for y in range (hauteur):
    for x in range (largeur):
        p=imageSource.getpixel((x,y))
        if(p<128):
            p=0
        else:
            p=255
        imageNB.putpixel((x,y),p)

imageNB.save("Dog.pbm")  #Le fichier image de sortie
```

➔ A votre avis, combien de fois parcourt-on la première puis la deuxième boucle for ? Détaillez votre réponse.
➔ Combien d'itérations parcourt-on au total dans ce programme ?
➔ Sous Python, écrire le programme python SeuillageImage1.py. sans l'exécuter.
➔ Lancer en même temps un chronomètre et l'exécution de votre programme. Combien de temps faut-il pour réaliser le traitement de l'image ?
➔ Refaire la même manipulation que précédemment avec les deux images suivantes :
Einstein.pgm et GirlBook.pgm (vous devez remplacer dog.pgm et dog.pbm)

➔ Remplir le tableau suivant :

Image à traiter	Taille image largeur	Taille image hauteur	Nombre totale d'itérations	Temps chronomètre
Dog.pgm				
Einstein.pgm				
GirlBook.pgm				

➔ Conclusion. Que constatez-vous ?
➔ Existe-t-il un lien entre le nombre de pixels à traiter et le temps de traitement ?

➔ Ecrire un second programme nommé SeuillageImage2.py réalisant un négatif.

Exercice 2 : (une suite à l'activité débranchée)

Travail demandé :

➔ Programmer en Python les fonctions qui prennent pour argument img une image (liste de liste de pixels) et k un entier .

Bord(img , k) qui renvoie la même image dont les k pixels du bord sont transformés en noir.
LigneV(Img , k) qui trace une ligne verticale de k pixels au milieu de l'image Img.
LigneH(Img , k) qui trace une ligne verticale de k pixels au milieu l'image Img.

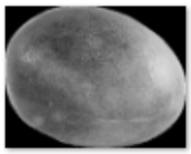
Exercice 3 : Quelques traitements usuels.

Travail demandé :

➔ On souhaite transformer une image couleur en une image de niveaux de gris. Programmer en python. La formule suivante permet de transformer un pixel couleur en pixel de niveau de gris.
NivPixelGris = IntensitéPixelRouge * 0.299 + IntensitéPixelVert * 0.587 + IntensitéPixelBleu ** 0.114)

Exemple :


mango.png


image_en_gris.png

Avant Après

➔ Ecrire un programme qui permet de réaliser un effet miroir (symétrie axiale par rapport à l'axe verticale)
Exemple :


mango.png


miroir.png

➔ Ecrire un programme permettant d'obtenir le résultat suivant :
➔ Quelle transformation avez-vous réalisé ?


mango.png


Renversement.png

Les codes sources :

Pour l'exercice 3 TP Programmation :

```
import sys, time
from PIL import Image

def ouvrir_Fichier(ImageFile):
    image = Image.open(ImageFile)
    return image

def Engris(ImageFile):
    image = ouvrir_Fichier(ImageFile)
    colonne,ligne = image.size
    data = image.load()
    for i in range(ligne):
        for j in range(colonne):
            pixel = data[j,i]
            #gris = int(0.2125 * pixel[0] + 0.7154 * pixel[1] +  0.0721 * pixel[2]) # OK aussi
            gris = int( pixel[0] * 0.299) + int( pixel[1] * 0.587) + int( pixel[2] ** 0.114)
            p = (gris,gris,gris)
            data[j,i] = p
    image.save('image_en_gris.png')
    image.close()

def miroir(ImageFile):
    image = ouvrir_Fichier(ImageFile)
    colonne,ligne = image.size
    imageF = Image.new(image.mode,image.size)
    for i in range(ligne):
        for j in range(colonne):
            pixel = image.getpixel((j,i))
            imageF.putpixel((colonne-j-1,i), pixel)
    imageF.save('miroir.png')
    image.close()

def renversement(ImageFile):
    image = ouvrir_Fichier(ImageFile)
    colonne,ligne = image.size
    imageF = Image.new(image.mode,image.size)
    for i in range(ligne):
        for j in range(colonne):
            pixel = image.getpixel((j,i))
            imageF.putpixel((j,ligne-i-1), pixel)
    imageF.save('Renversement.png')
    image.close()

# Tests
ImageFile = 'mango.png'
print('\nVeullez consulter les images transformées dans votre répertoire courant\n')
Engris(ImageFile)
miroir(ImageFile)
Engris(ImageFile)
renversement(ImageFile)
```