Opérations élémentaires sur les images

S1-3 Images et Tableaux - Partie II

Lycée Carnot - Dijon, 2021 - 2022

Germain Gondor

Sommaire

1. Opérations sur les couleurs
2. Opérations de symétrie
3. Changement d'échelle

Objectifs

A la fin de la séquence d'enseignement les élèves doivent être capable :

- de modifier les couleurs d'une images :
  - niveaux de gris
  - noir et blanc
  - inversion des couleurs, ...
- d'obtenir les images miroirs vertical ou horizontal
- de faire tourner l'image de 90° ou 180°
- réduire ou agrandir une image d'un facteur r entier

Sommaire

1. Opérations sur les couleurs
   - Extraction d'un canal - fusion de canaux
   - Niveaux de gris
   - Image en noir et blanc
   - Inverse des couleurs
2. Opérations de symétrie
3. Changement d'échelle

Image de base

Extraction d'un canal

Version Rouge

Extraction d'un canal

Version Verte

Extraction d'un canal

Version Bleu

Extraction d'un canal

```
def extraction(T):
    R, G, B = [], [], []




    return R, G, B
```

Extraction d'un canal

```
def extraction(T):
    R, G, B = [], [], []
    for lig in T:
        R.append([])
        G.append([])
        B.append([])
        for pix in lig:
            R[-1].append([pix[0], 0, 0])
            G[-1].append([0, pix[1], 0])
            B[-1].append([0, 0, pix[2]])
    return R, G, B
```

Fusion de canaux

```
def fusion_canaux(R, G, B):
    T = []



    return T
```

```
image_somme = np.array(image_R) + np.array(image_G) + np.array(image_B)
```

Fusion de canaux

```
def fusion_canaux(R, G, B):
    T = []
    for i in range(len(R)):
        T.append([])
        for j in range(len(R[0])):
            T[-1].append([R[i][j][0], G[i][j][1], B[i][j][2]])
    return T
```

Fusion de canaux

```
def fusion_canaux(R, G, B):
    T = []
    for i in range(len(R)):
        T.append([])
        for j in range(len(R[0])):
            T[-1].append([R[i][j][0], G[i][j][1], B[i][j][2]])
    return T
```

```
image_somme = np.array(image_R) + np.array(image_G) + np.array(image_B)
```

Niveaux de gris

Moyenne géométrique

Niveaux de gris

Moyenne arithmétique

Niveaux de gris

21,26% rouge, 71,52% vert et 7,22% bleu

Niveaux de gris

```
def niveau_gris_geom(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:


    return res
```

Niveaux de gris

```
def niveau_gris_geom(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            moy = int(np.sqrt(pix[0]**2+pix[1]**2+pix[2]**2)/np.sqrt(3))
            res[-1].append([moy, moy, moy])
    return res
```

Niveaux de gris

```
def niveau_gris_geom(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            moy = int(np.sqrt(pix[0]**2+pix[1]**2+pix[2]**2)/np.sqrt(3))
            res[-1].append([moy, moy, moy])
    return res
```

```
moy = int((pix[0]/3+pix[1]/3+pix[2]/3))
```

Niveaux de gris

```
def niveau_gris_geom(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            moy = int(np.sqrt(pix[0]**2+pix[1]**2+pix[2]**2)/np.sqrt(3))
            res[-1].append([moy, moy, moy])
    return res
```

```
moy = int((pix[0]/3+pix[1]/3+pix[2]/3))
```

```
moy = int((pix[0]*0.2126+pix[1]*0.7152+pix[2]*0.0722))
```

Image en noir et blanc

Seuil à 30

Image en noir et blanc

Seuil à 60

Image en noir et blanc

Seuil à 90

Image en noir et blanc

Seuil à 120

Image en noir et blanc

Seuil à 150

Image en noir et blanc

Seuil à 180

Image en noir et blanc

Seuil à 210

Image en noir et blanc

Seuil à 240

Image en noir et blanc

```
def noiretblanc(T, seuil):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:









    return res
```

Image en noir et blanc

```
def noiretblanc(T, seuil):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            moy = int(np.sqrt(pix[0]**2+pix[1]**2+pix[2]**2)/np.sqrt(3))
            if moy > seuil:
                res[-1].append([255, 255, 255])
            else:
                res[-1].append([0, 0, 0])
    return res
```

Inverse des couleurs

Inverse des couleurs

```
def inverse(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            res[-1].append(
    return res
```

Inverse des couleurs

```
def inverse(T):
    res = []
    for lig in T:
        res.append([])
        for pix in lig:
            res[-1].append([255-pix[0], 255-pix[1], 255-pix[2]])
    return res
```

Sommaire

1 Opérations sur les couleurs

2 Opérations de symétrie
- Miroirs horizontaux et verticaux
- Rotations

3 Changement d'échelle

Miroir horizontal

Miroir vertical

Miroirs horizontaux et verticaux

```
def miroir_v(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)




    return res
```

```
def miroir_h(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)




    return res
```

Miroirs horizontaux et verticaux

```
def miroir_v(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)
    for i in range(nl):
        for j in range(nc):
            for k in range(3):
                res[i][j][k] = T[i][nc -1 -j][k]
    return res
```

```
def miroir_h(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)




    return res
```

Miroirs horizontaux et verticaux

```
def miroir_v(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)
    for i in range(nl):
        for j in range(nc):
            for k in range(3):
                res[i][j][k] = T[i][nc -1 -j][k]
    return res
```

```
def miroir_h(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)
    for i in range(nl):
        for j in range(nc):
            for k in range(3):
                res[i][j][k] = T[nl -1 -i][j][k]
    return res
```

Rotations de 180°

Rotations de 90°

Rotations de 180°

```
def rotation_180(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)




    return res
```

Rotations de 180°

```
def rotation_180(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nl, nc, 3), dtype =np.uint8)
    for i in range(nl):
        for j in range(nc):
            for k in range(3):
                res[i][j][k] = T[nl -1 -i][nc -1 -j][k]
    return res
```

Rotations d'un quart de tour

```
def rotation_90(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nc, nl, 3), dtype =np.uint8)



    return res
```

Rotations d'un quart de tour

```
def rotation_90(T):
    nl, nc = len(T), len(T[0])
    res = np.zeros((nc, nl, 3), dtype =np.uint8)
    for i in range(nl):
        for j in range(nc):
            for k in range(3):
                res[j][i][k] = T[nl -1 -i][j][k]
    return res
```

Sommaire

1 Opérations sur les couleurs

2 Opérations de symétrie

3 Changement d'échelle
- Réduction
- Agrandissement

Image de base

Réduction

d'un facteur 2

Réduction

d'un facteur 4

Réduction

d'un facteur 6

Réduction

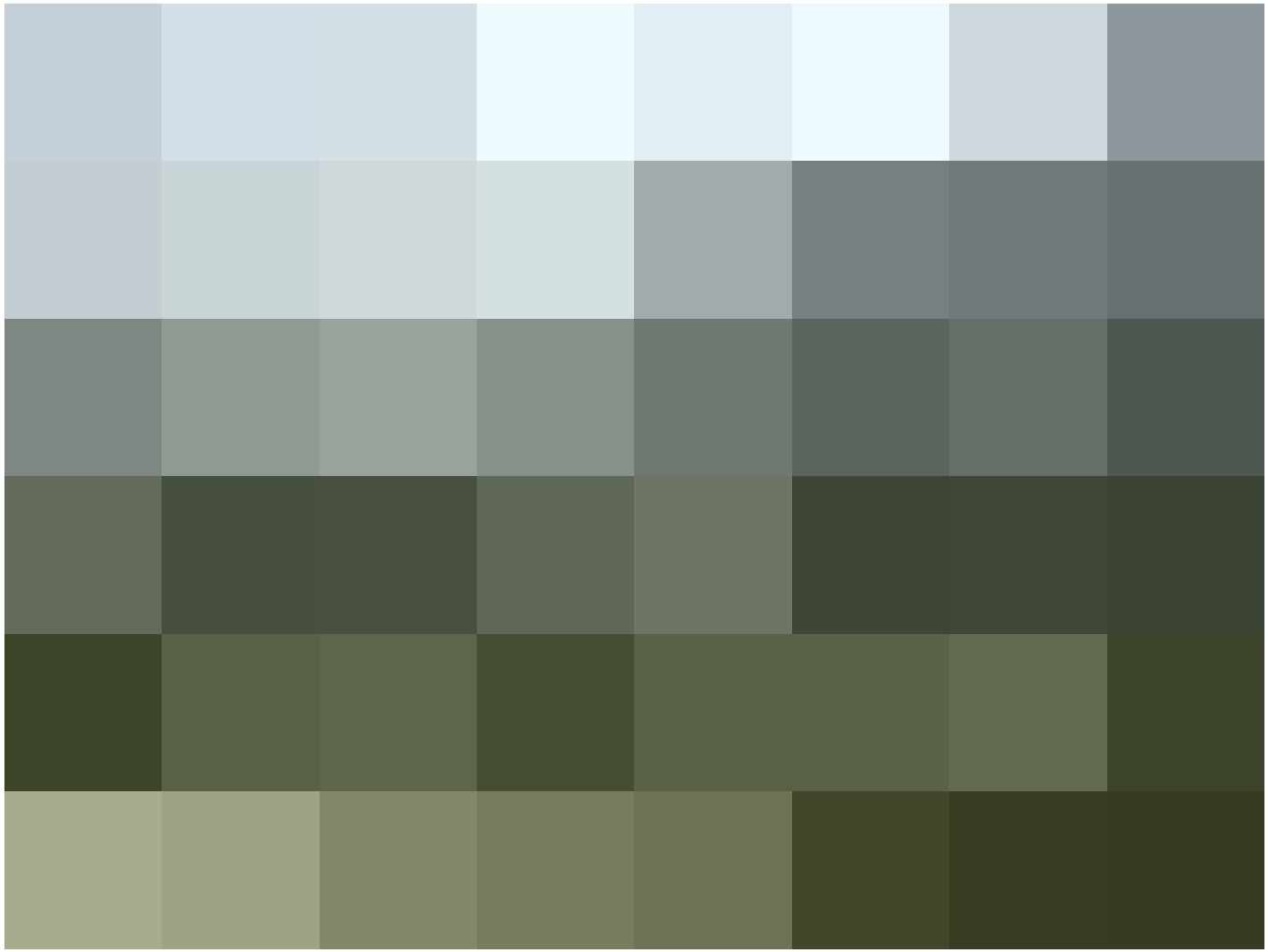
d'un facteur 10

Réduction

d'un facteur 20

Réduction

d'un facteur 50



Réduction

```
def reduction(T, r):
    """ Réduit une image d'un facteur entier r """




    return res
```

Réduction

```
def reduction(T, r):
    """ Réduit une image d'un facteur entier r """
    nl, nc = len(T), len(T[0])
    rnl, rnc = nl//r, nc//r
    res = np.zeros((rnl, rnc, 3), dtype = np.uint8)
    for i in range(rnl):
        for j in range(rnc):
            moy_R, moy_G, moy_B = 0, 0, 0
            for p in range(r):
                for q in range(r):
                    moy_R += T[i*r+p][j*r+q][0]
                    moy_G += T[i*r+p][j*r+q][1]
                    moy_B += T[i*r+p][j*r+q][2]
            moy_R = moy_R // (r**2)
            moy_G = moy_G // (r**2)
            moy_B = moy_B // (r**2)

            res[i][j][0] = moy_R
            res[i][j][1] = moy_G
            res[i][j][2] = moy_B
    return res
```

Agrandissement

```
def agrandissement(T, r):
    nl, nc = len(T), len(T[0])
    rnl, rnc = nl*r, nc*r
    res = np.zeros((rnl, rnc, 3), dtype = np.uint8)
    for i in range(nl):
        for j in range(nc):
            for p in range(r):
                for q in range(r):
                    for k in range(3):
                        res[i*r+p][j*r+q][k] = T[i][j][k]
    return res
```

Agrandissement

```
def agrandissement(T, r):




    return res
```