TIPE

Comment homogénéiser et traiter des images afin de les diffuser ?

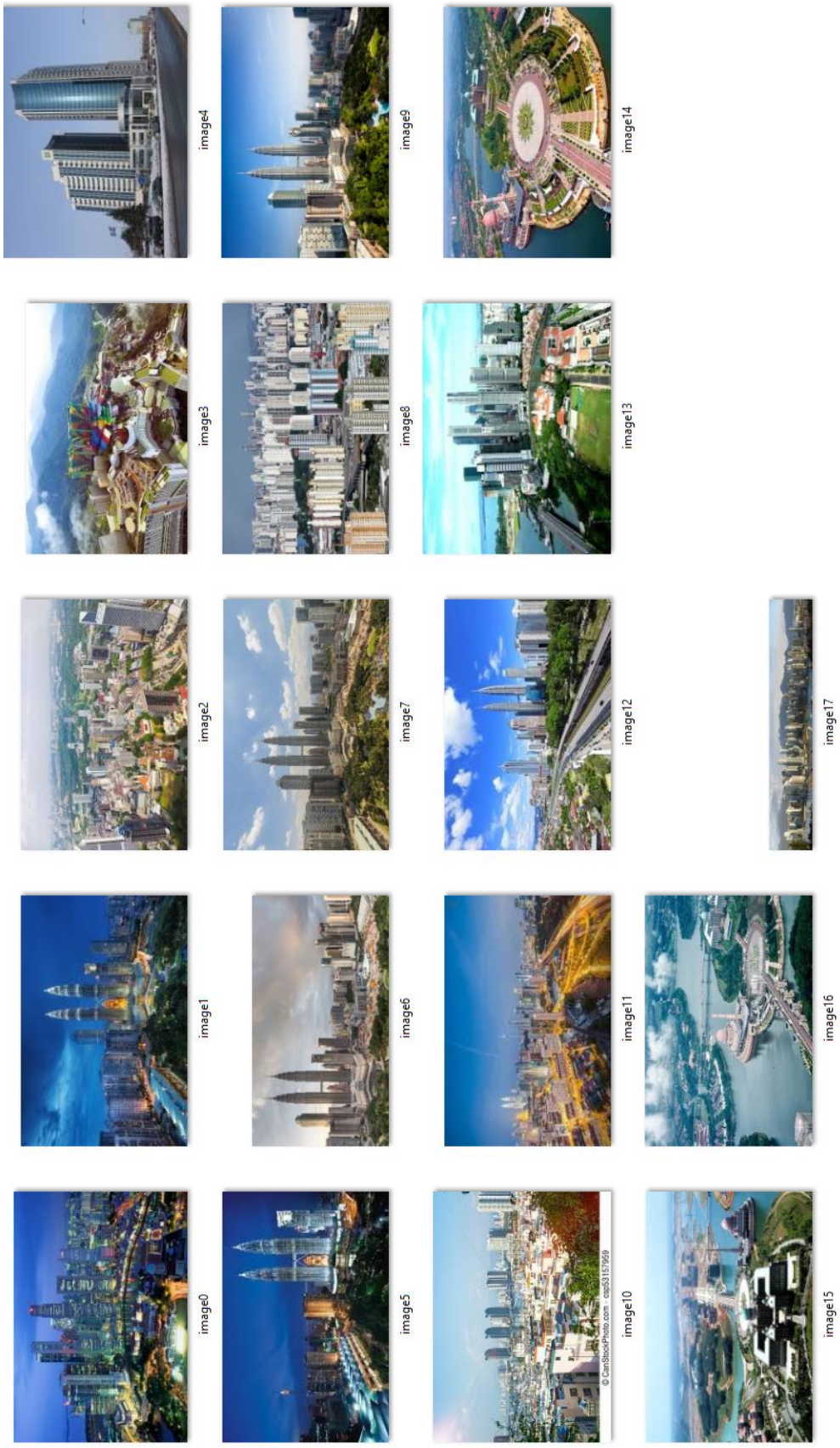
Annexes

Clément LAGNEAU

Sommaire :

I. Banques d'image :

a. Banque ville :



b. Banque salon :


image4


image3


image2


image1


image6


image0


image5

II. Codes python :

a. Moyenne image

```
from PIL import Image

def moyenne_image(img):
    image = Image.open(img) # on ouvre l'image
    enMemoire = image.load() # on la charge en memoire
    colonne,ligne = image.size #on calcul le nbre de pixel
    n = colonne*ligne
    moy_r=0
    moy_v=0
    moy_b=0
    for i in range(colonne):
        for j in range(ligne):
            p=enMemoire[i,j]
            moy_r += p[0]
            moy_v += p[1]
            moy_b += p[2]
    image.close()
    return((moy_r/n,moy_v/n,moy_b/n))
```

b. Coefficient de contours

```
from PIL import Image

def coef(URLImage):
    image = Image.open(URLImage)
    colonne,ligne = image.size
    n=colonne*ligne
    image_en_memoire = image.load()
    somme=0
    for x in range(colonne):
        for y in range(ligne):
            p = image_en_memoire[x,y]
            if p==(255,255,255) :
                somme += 1
    return(somme/n)

```

c. Convolution pour détection des contours

```
from PIL import Image

Filtre = [[-1,-1,-1],
          [-1,8,-1],
          [-1,-1,-1]]

def convolution(URLImage, Filtre):
    def Convolution(Filtre,image,x,y):
        p0 = 0
        p1 = 0
        p2 = 0
        for i in range(-1,2):
            for j in range(-1,2):
                p0 += Filtre[i+1][j+1]*image[x+i,y+j][0]
                p1 += Filtre[i+1][j+1]*image[x+i,y+j][1]
                p2 += Filtre[i+1][j+1]*image[x+i,y+j][2]
                # normalisation des composantes
                p0 = int(p0)
                p1 = int(p1)
                p2 = int(p2)
        # retourne le pixel convolué
        return (p0,p1,p2)

    image = Image.open(URLImage)
    resultat = Image.new("RGB", image.size , "white")
    colonne,ligne = image.size
    image_en_memoire = image.load()
    res = resultat.load()
    for x in range(1,colonne-1):
        for y in range(1,ligne-1):
            p = Convolution(Filtre,image_en_memoire,x,y)
            res[x, y]=p
    resultat.save('convolution.'+URLImage)
    resultat.close()
    image.close()


def contours(URLImage, resolution = 100):
    image = Image.open(URLImage)
    colonne,ligne = image.size
    image_en_memoire = image.load()
    for x in range(colonne):
        for y in range(ligne):
            p = image_en_memoire[x,y]
            m=p[0]+p[1]+p[2]
            m = int(m/3)
            if m>resolution :
                image_en_memoire[x,y] = (255,255,255)
            else:
                image_en_memoire[x,y] = (0,0,0)
    image.save('contours.'+URLImage)


def detection_contours(URLImage):
    convolution(URLImage, Filtre)
    contours('convolution.'+URLImage)
```

d. Application à toute la banque d'image

```
import os
monRepertoire='D:/Cours/MP/TIPE/Banque_image/ville'
os.chdir(monRepertoire)
fichiers = [f for f in os.listdir(monRepertoire) if os.path.isfile(os.path.join(monRepertoire, f))]

dic={}
somme=[0,0,0]
coefficient = 0
indice=0

for image in fichiers:
    indice += 1

    moy = moyenne_image(image)
    somme[0] += moy[0]
    somme[1] += moy[1]
    somme[2] += moy[2]

    dic[image]=moy

    detection_contours(image)
    coefficient += coef('contours.convolution.'+image)

somme[0] = somme[0]/indice
somme[1] = somme[1]/indice
somme[2] = somme[2]/indice
coefficient = coefficient/indice

print(somme,coefficient)

resolution = 0.1
if not os.path.exists('defectueuses'):
    os.makedirs('defectueuses')


def modif_moyenne(URLImage,moyimage,moybanque):
    image = Image.open(URLImage)
    enMemoire = image.load()
    colonne,ligne = image.size
    r=int(moybanque[0]-moyimage[0])
    v=int(moybanque[1]-moyimage[1])
    b=int(moybanque[2]-moyimage[2])
    p0=0
    p1=0
    p2=0
    for i in range(colonne):
        for j in range(ligne):
            p=enMemoire[i,j]
            p0 = p[0] + r
            p1 = p[1] + v
            p2 = p[2] + b
            enMemoire[i,j]=(p0,p1,p2)
    image.save('moyenne_modifie.'+URLImage)
    image.close()

for image in fichiers:
    if abs(coef(image)-coefficient)>resolution:
        os.rename(image,monRepertoire+'/defectueuses/'+image)
    else:
        modif_moyenne(image,dic[image],somme)

```

e. Convolution n

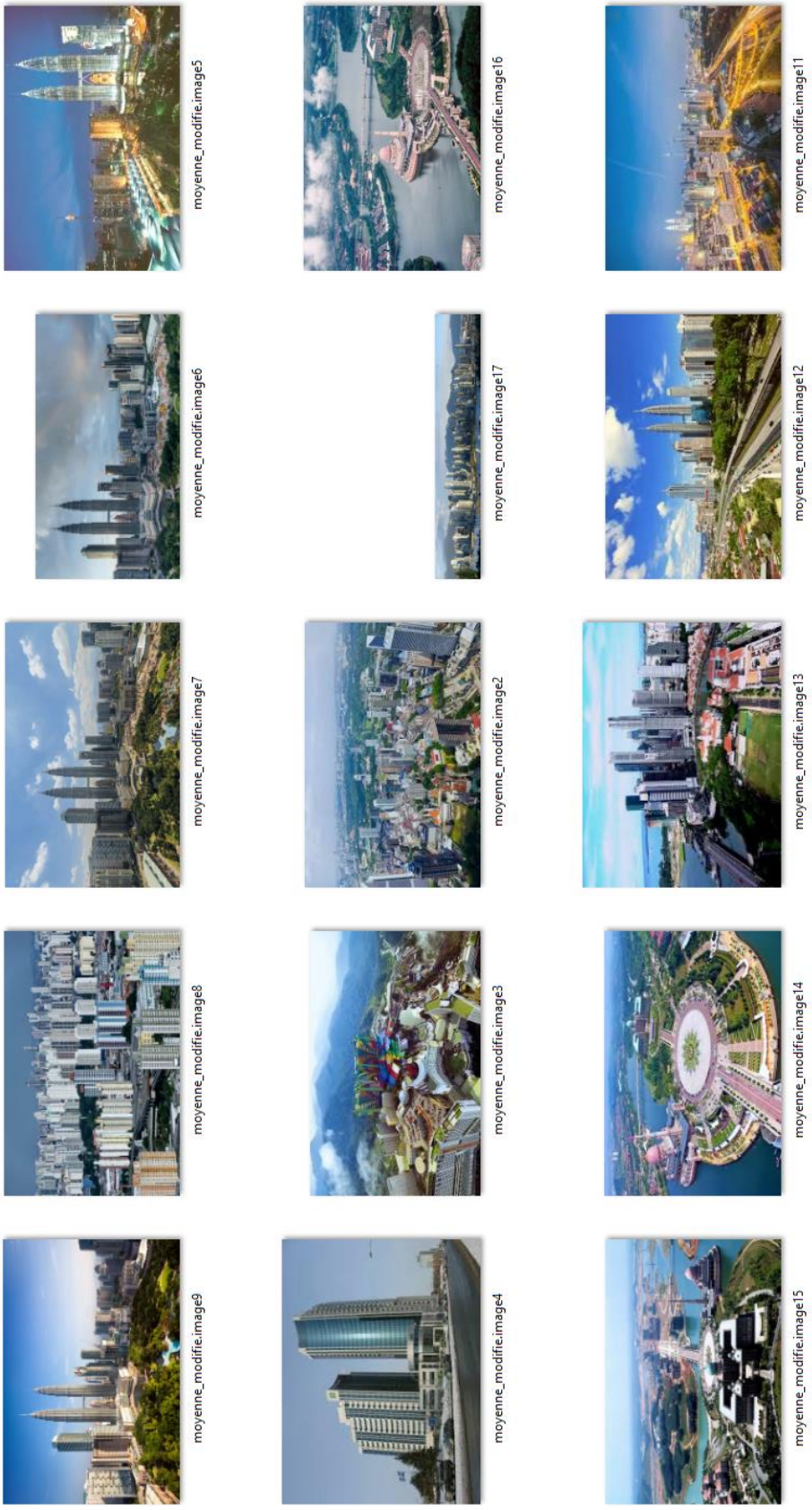
```
def filtre_test2():
    res=[[x for x in range(-n+k+1,k+1)]for k in range(n)]
    return(res)

from PIL import Image

#n impair
n=3
part=n//2

def creation_filtre():
    res=[[1/n**2 for x in range(n)]for x in range(n)]
    return(res)

URLImage = 'image.jpg' #Pointe vers l'image

Filtre = creation_filtre()


def Convolution(Filtre,image,x,y):
  p0 = 0
  p1 = 0
  p2 = 0
  for i in range(-part,part+1):
      for j in range(-part,part+1):
          p0 += Filtre[i+part][j+part]*image[x+i,y+j][0]
          p1 += Filtre[i+part][j+part]*image[x+i,y+j][1]
          p2 += Filtre[i+part][j+part]*image[x+i,y+j][2]
          # normalisation des composantes
          p0 = int(p0)
          p1 = int(p1)
          p2 = int(p2)
          # retourne le pixel convolué
  return (p0,p1,p2)

def convolutionne(Filtre, URLImage):
    #Ouverture Image
    try:
        image = Image.open(URLImage)
    except IOError:
        print ('Erreur sur ouverture du fichier ')
    #Declaration des variables
    resultat = Image.new("RGB", image.size , "black")
    colonne,ligne = image.size
    image_en_memoire = image.load()
    #Traitement
    for y in range(part,ligne-part):
        for x in range(part,colonne-part):
            p = Convolution(Filtre,image_en_memoire,x,y)
            resultat.putpixel((x,y),p)
    #On enregistre l'image
    resultat.save("Convolution-"+str(n)+"-"+URLImage)
    # fermeture du fichier image
    resultat.close()
    image.close()


convolutionne(Filtre, URLImage)
```

f. Contours OpenCV

```
import cv2
from PIL import Image


img = cv2.imread('image_opencv.jpg',0)
ret,thresh = cv2.threshold(img,127,255,0)
image, contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
img2 = cv2.drawContours(img, contours, -1, (0,255,0), 3)
cv2.namedWindow('image', cv2.WINDOW_NORMAL)
cv2.imshow('image',img2)
k = cv2.waitKey(0)
if k == 27:
    cv2.destroyAllWindows()
cv2.imwrite('image_opencv_modifie.jpg',img2)


def contours(URLImage, resolution = 100):
    img0 = Image.open(URLImage)
    colonne,ligne = img0.size
    image_en_memoire = img0.load()
    img = Image.new("RGB", img0.size , "white")
    res = img.load()
    for x in range(1,colonne-1):
        for y in range(1,ligne-1):
            p = image_en_memoire[x,y]
            if p>resolution :
                res[x,y] = (255,255,255)
            else:
                res[x,y] = (0,0,0)
    img.save(('.'.join(URLImage.split('.')[:-1]))+".contours."+(URLImage.split('.')[-1]))

def coef(URLImage):
    image = Image.open(URLImage)
    colonne,ligne = image.size
    n=colonne*ligne
    image_en_memoire = image.load()
    somme=0
    for x in range(1,colonne-1):
        for y in range(1,ligne-1):
            p = image_en_memoire[x,y]
            if p==(255,255,255):
                somme += 1
    return(somme,n)

```

III. Résultats sur la banque d'image

a. Banque ville :



moyenne_modifie.image10


moyenne_modifie.image1


moyenne_modifie.image0


contours.convolution.image9


convolution.image9


contours.convolution.image8


convolution.image8


contours.convolution.image7


convolution.image7


contours.convolution.image6


convolution.image6


contours.convolution.image5


convolution.image5


contours.convolution.image4


convolution.image4


contours.convolution.image3


convolution.image3


contours.convolution.image2


convolution.image2


contours.convolution.image17

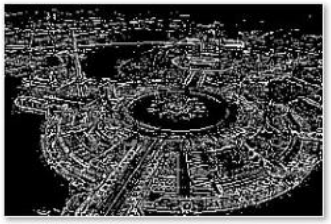

convolution.image17


contours.convolution.image16


convolution.image16


contours.convolution.image15


convolution.image15


contours.convolution.image14


convolution.image14


contours.convolution.image13


convolution.image13


contours.convolution.image12


convolution.image12


contours.convolution.image11


convolution.image11


contours.convolution.image10


convolution.image10


contours.convolution.image1


convolution.image1


contours.convolution.image0


convolution.image0


image16

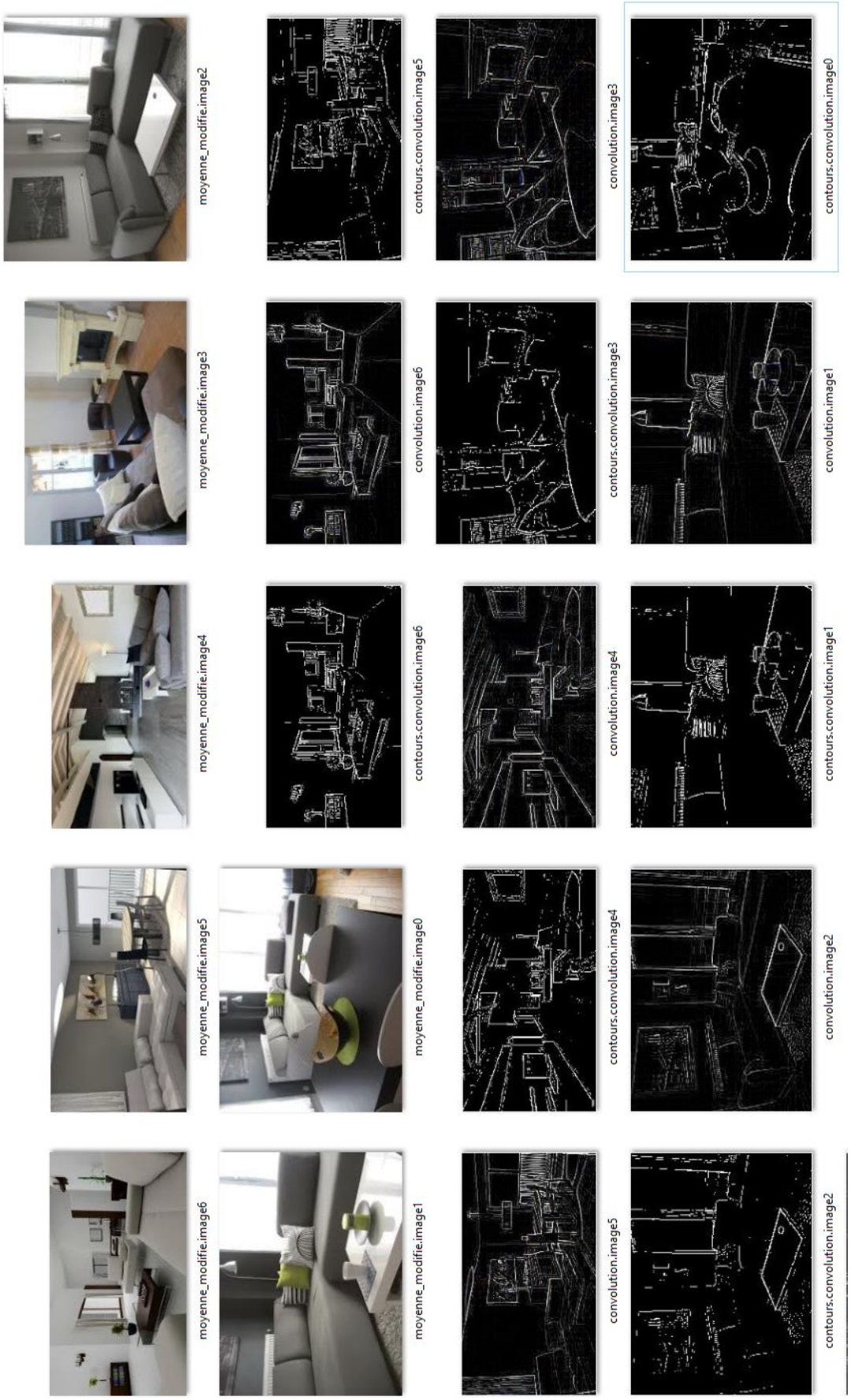
dic - Dictionnaire (18 éléments)

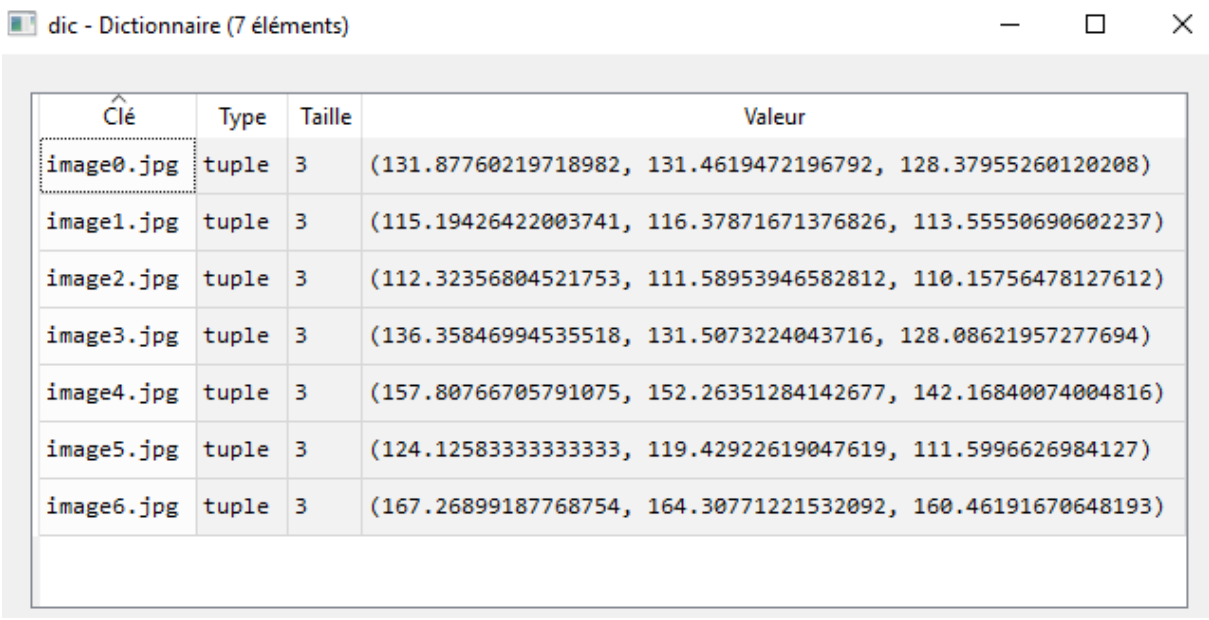
Clé	Type	Taille	Valeur
image0.jpg	tuple	3	(72.14741843975128, 93.85250762181973, 124.02117596606719)
image1.jpg	tuple	3	(46.43680079483358, 82.44500745156483, 122.4497565822156)
image10.jpg	tuple	3	(157.17688194444443, 167.02274305555557, 166.7158263888889)
image11.jpg	tuple	3	(96.7397516145057, 114.32890213611525, 128.17629408842524)
image12.jpg	tuple	3	(108.97097489506613, 135.3884335154827, 169.48984319315753)
image13.jpg	tuple	3	(132.21850495561836, 160.92049118337778, 150.12142260080404)
image14.jpg	tuple	3	(120.11573122529644, 122.9049604743083, 115.19966403162056)
image15.jpg	tuple	3	(121.00953781079129, 140.44350046425257, 143.45862478076964)
image16.jpg	tuple	3	(105.65090349865436, 139.27794998718443, 149.99597510572858)
image17.jpg	tuple	3	(127.90036496350365, 131.96922445255476, 130.82762773722627)
image2.jpg	tuple	3	(150.37375062096373, 150.92147044212618, 147.3653253849975)
image3.jpg	tuple	3	(132.88934523809525, 136.90410714285716, 139.40789682539682)
image4.jpg	tuple	3	(117.33117064044899, 131.3114277753453, 152.51122477411138)
image5.jpg	tuple	3	(49.732081470442125, 77.15888723298559, 112.87846994535519)
image6.jpg	tuple	3	(146.13452086840067, 142.09606614709483, 137.24022572829352)
image7.jpg	tuple	3	(130.8098927492648, 137.26657662572686, 138.35527471357668)
image8.jpg	tuple	3	(139.5880662020906, 144.13591225847324, 147.71940133037694)
image9.jpg	tuple	3	(103.31545416666667, 126.171753125, 139.59610520833334)

Somme = [114.36339733882434, 129.69555117176805, 139.75167413251918]

Coefficient = 0.09082417479874944

b. Banque salon :





Clé	Type	Taille	Valeur
image0.jpg	tuple	3	(131.87760219718982, 131.4619472196792, 128.37955260120208)
image1.jpg	tuple	3	(115.19426422003741, 116.37871671376826, 113.55550690602237)
image2.jpg	tuple	3	(112.32356804521753, 111.58953946582812, 110.15756478127612)
image3.jpg	tuple	3	(136.35846994535518, 131.5073224043716, 128.08621957277694)
image4.jpg	tuple	3	(157.80766705791075, 152.26351284142677, 142.16840074004816)
image5.jpg	tuple	3	(124.12583333333333, 119.42922619047619, 111.5996626984127)
image6.jpg	tuple	3	(167.26899187768754, 164.30771221532092, 160.46191670648193)

Somme = [134.9937709538188, 132.4197110072673, 127.77268914374575]

Coefficient = 0.04473163693839776