Comment homogénéiser et traiter des images afin de les diffuser ?

LAGNEAU Clément

Sommaire

- Introduction
- Les images
- Traitement d'une image
- Caractérisation d'une image
- Traitement pour la banque d'image
- Conclusion

Introduction







Banque d'Images

Caractérisation de chaque image

Banque d'Images

Traitement des images

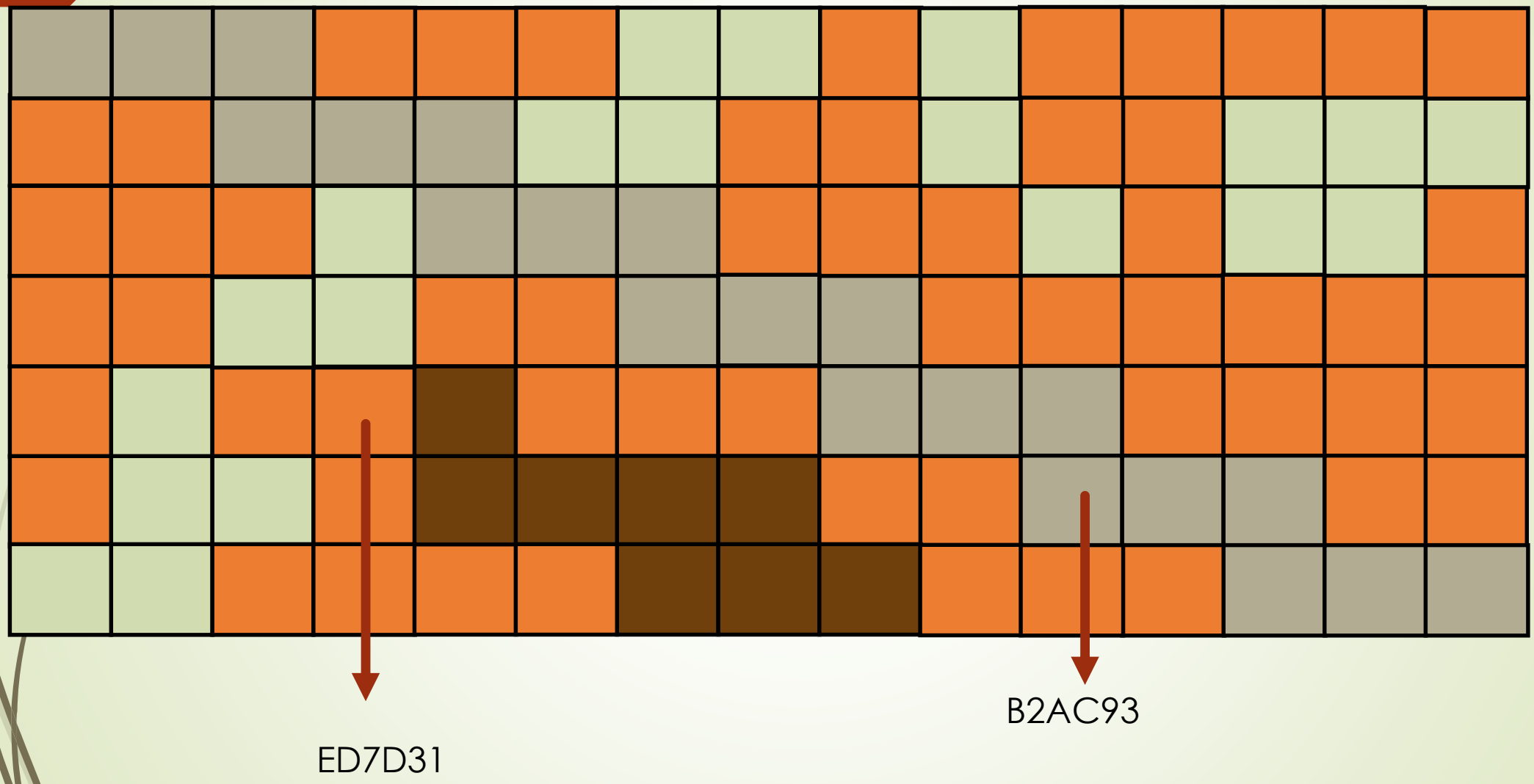
Banque d'Images homogène

(Images supprimées)

- Commerçant
- Suivi d'une charte graphique

Les images

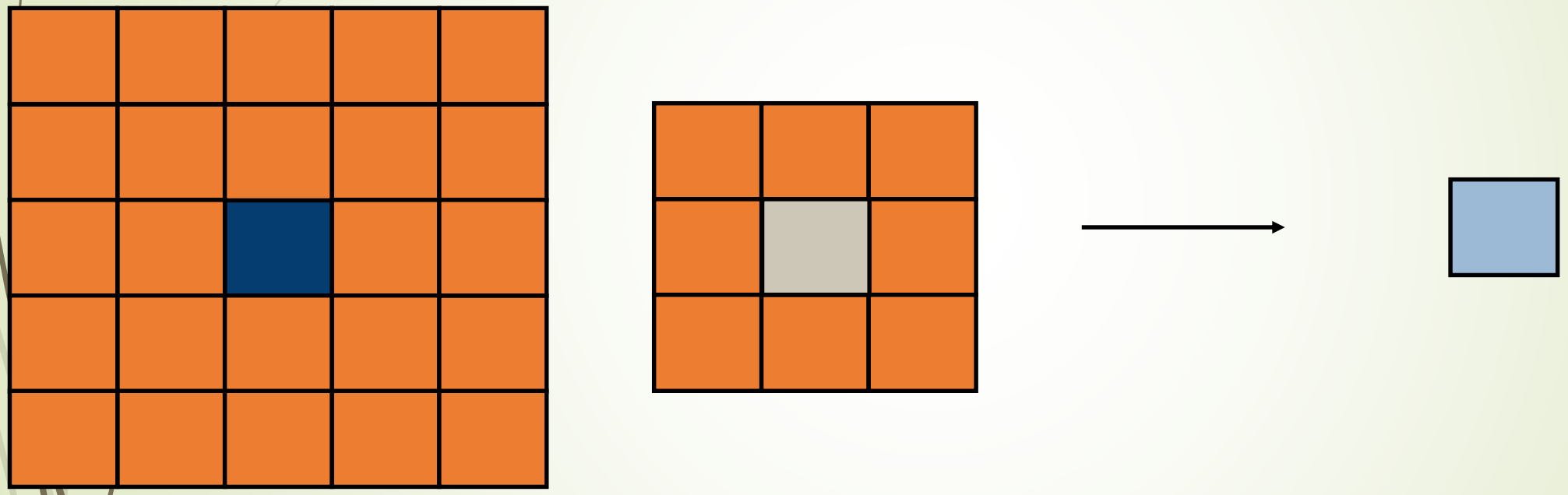
- Ensemble de pixels
- 3 octets, 1 Rouge, 1 Vert, 1 Bleu
- 256 possibilités, 256*256*256 = 16 777 216 couleurs possibles
- 1 mégapixel : 1280 x 960
- 4 mégapixels : 2400 x 1800
- 8 mégapixels : 3200 x 2400
- 12 mégapixels : 4000 x 3000

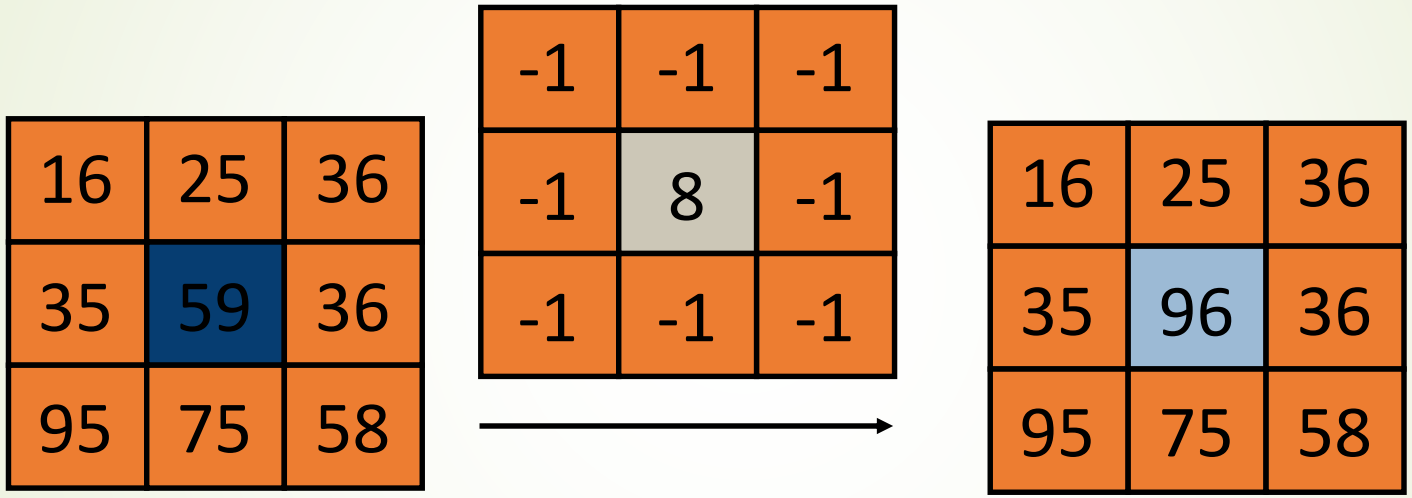

ED7D31
B2AC93

Traitement d'image

- Filtre de convolution
- Module PIL de Python (Python Imaging Library)
- Module OpenCV

Filtre de convolution




16 25 36
35 59 36
95 75 58
-1 -1 -1
-1 8 -1
-1 -1 -1
16 25 36
35 96 36
95 75 58

Quelques filtres classiques

-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacien

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Masques de Prewitt

1/3	1/3	1/3
1/3	1/3	1/3
1/3	1/3	1/3

Floue

Présentation du module Image de PIL

```
 8 from PIL import Image
 9
10 def negatif(URLImage):
11     image = Image.open(URLImage)
12     colonne,ligne = image.size
13     res = Image.new("RGB", image.size , "white")
14     image_en_memoire = image.load()
15     p0=0
16     p1=0
17     p2=0
18     for y in range(ligne):
19         for x in range(colonne):
20             p0 = 255-image_en_memoire[x,y][0]
21             p1 = 255-image_en_memoire[x,y][1]
22             p2 = 255-image_en_memoire[x,y][2]
23             res.putpixel((x,y),(p0,p1,p2))
24     res.save("negatif-"+URLImage)
25     res.close()
26     image.close()
27
```

Négatif

Impact sur la grandeur du filtre

```
22
23 from PIL import Image
24
25 #n impair
26 n=3
27 part=n//2
28
29 def creation_filtre():
30     res=[[1/n**2 for x in range(n)]for x in range(n)]
31     return(res)
32
33 URLImage = 'image.jpg' #Pointe vers l'image
34
35 Filtre = creation_filtre()
36
```

```
37
38 def Convolution(Filtre,image,x,y):
39   p0 = 0
40   p1 = 0
41   p2 = 0
42   for i in range(-part,part+1):
43       for j in range(-part,part+1):
44           p0 += Filtre[i+part][j+part]*image[x+i,y+j][0]
45           p1 += Filtre[i+part][j+part]*image[x+i,y+j][1]
46           p2 += Filtre[i+part][j+part]*image[x+i,y+j][2]
47           # normalisation des composantes
48           p0 = int(p0)
49           p1 = int(p1)
50           p2 = int(p2)
51           # retourne le pixel convolué
52   return (p0,p1,p2)
53
```

```
54 def convolutionne(Filtre, URLImage):
55     #Ouverture Image
56     try:
57         image = Image.open(URLImage)
58     except IOError:
59         print ('Erreur sur ouverture du fichier ')
60     #Declaration des variables
61     resultat = Image.new("RGB", image.size , "black")
62     colonne,ligne = image.size
63     image_en_memoire = image.load()
64     #Traitement
65     for y in range(part,ligne-part):
66         for x in range(part,colonne-part):
67             p = Convolution(Filtre,image_en_memoire,x,y)
68             resultat.putpixel((x,y),p)
69     #On enregistre l'image
70     resultat.save("Convolution-"+str(n)+"-"+URLImage)
71     # fermeture du fichier image
72     resultat.close()
73     image.close()
74
75
76 convolutionne(Filtre, URLImage)
77
```

Evolution en fonction du filtre

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1

3

5

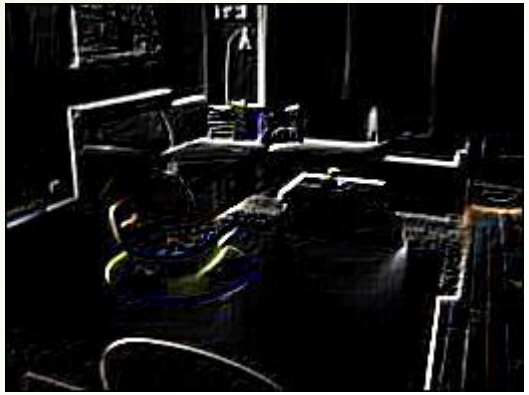
7

9

-2	-1	0
-1	0	1
0	1	2

1

3



5

7

9

15

OpenCV

- Module python spécialisé dans le traitement d'image

Caractérisation d'une image

- Moyenne des couleurs
- Contours

Moyenne des couleurs

```
 8
 9 from PIL import Image
10
11 def moyenne_image(img):
12     image = Image.open(img) # on ouvre l'image
13     enMemoire = image.load() # on la charge en memoire
14     colonne,ligne = image.size #on calcul le nbre de pixel
15     n = colonne*ligne
16     moy_r=0
17     moy_v=0
18     moy_b=0
19     for i in range(colonne):
20         for j in range(ligne):
21             p=enMemoire[i,j]
22             moy_r += p[0]
23             moy_v += p[1]
24             moy_b += p[2]
25     image.close()
26     return((moy_r/n,moy_v/n,moy_b/n))
27
```

(112, 111, 110)

(115, 116, 113)

(167, 164, 160)

Détection des contours

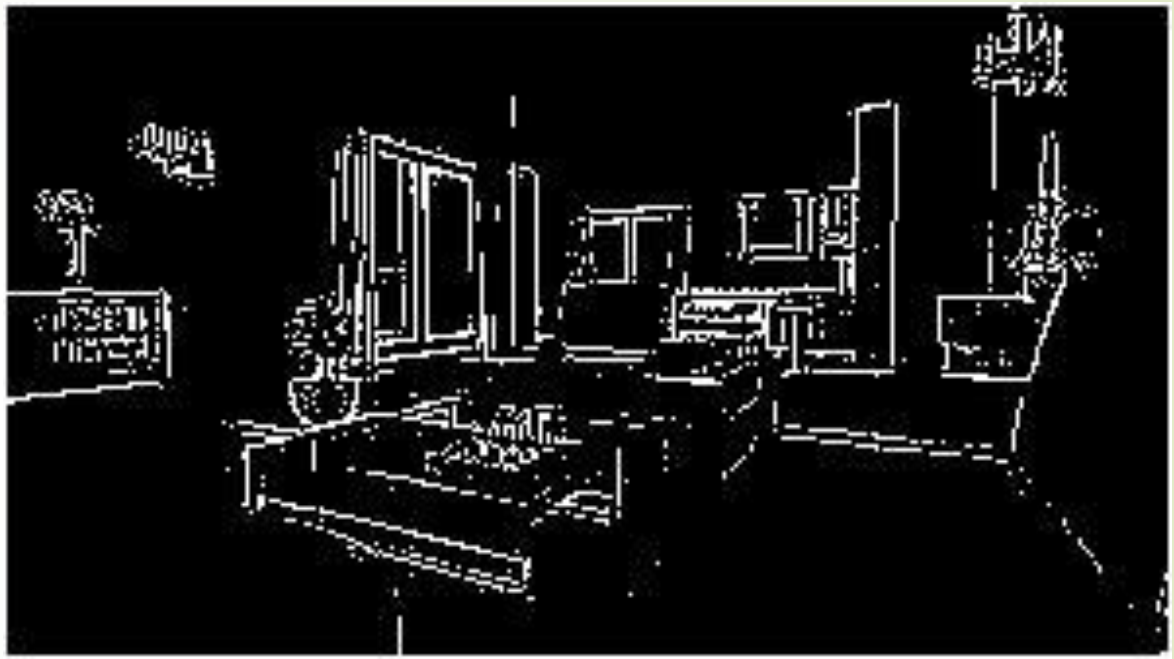
-1	-1	-1
-1	8	-1
-1	-1	-1

```
25
26 def convolution(URLImage, Filtre):
27     def Convolution(Filtre,image,x,y):
28         p0 = 0
29         p1 = 0
30         p2 = 0
31         for i in range(-1,2):
32             for j in range(-1,2):
33                 p0 += Filtre[i+1][j+1]*image[x+i,y+j][0]
34                 p1 += Filtre[i+1][j+1]*image[x+i,y+j][1]
35                 p2 += Filtre[i+1][j+1]*image[x+i,y+j][2]
36                 # normalisation des composantes
37                 p0 = int(p0)
38                 p1 = int(p1)
39                 p2 = int(p2)
40         # retourne le pixel convolué
41         return (p0,p1,p2)
42
43     image = Image.open(URLImage)
44     resultat = Image.new("RGB", image.size , "white")
45     colonne,ligne = image.size
46     image_en_memoire = image.load()
47     res = resultat.load()
48     for x in range(1,colonne-1):
49         for y in range(1,ligne-1):
50             p = Convolution(Filtre,image_en_memoire,x,y)
51             res[x, y]=p
52     resultat.save('convolution.'+URLImage)
53     resultat.close()
54     image.close()
55
```

Convolution

```
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)

```

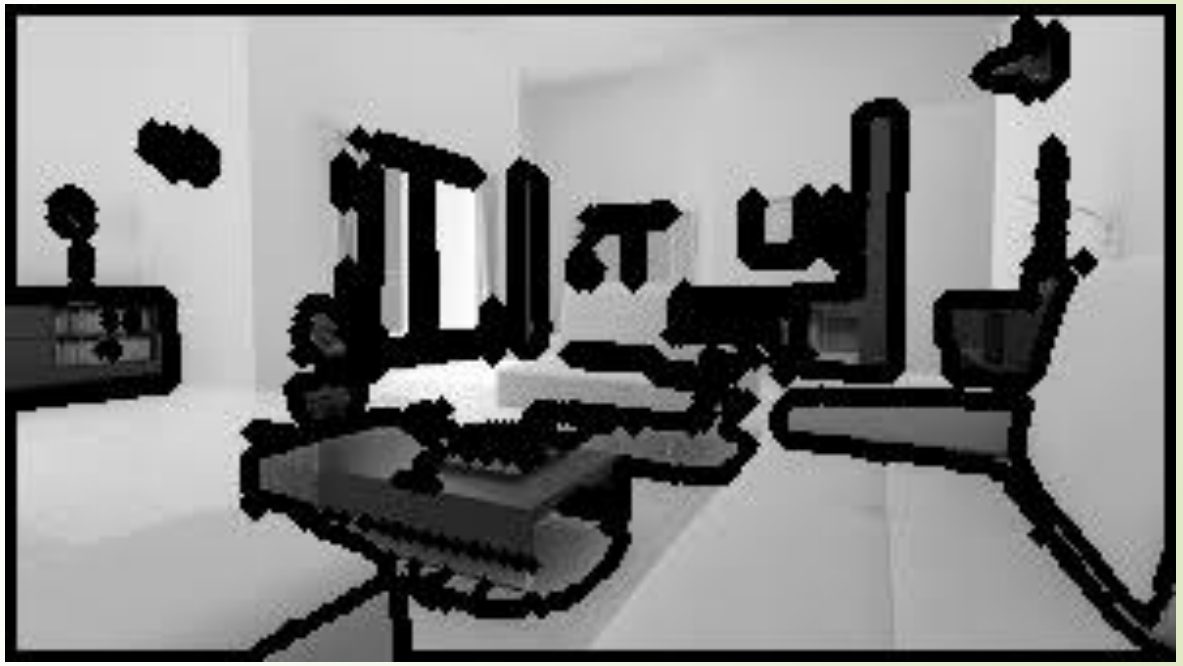
Contours



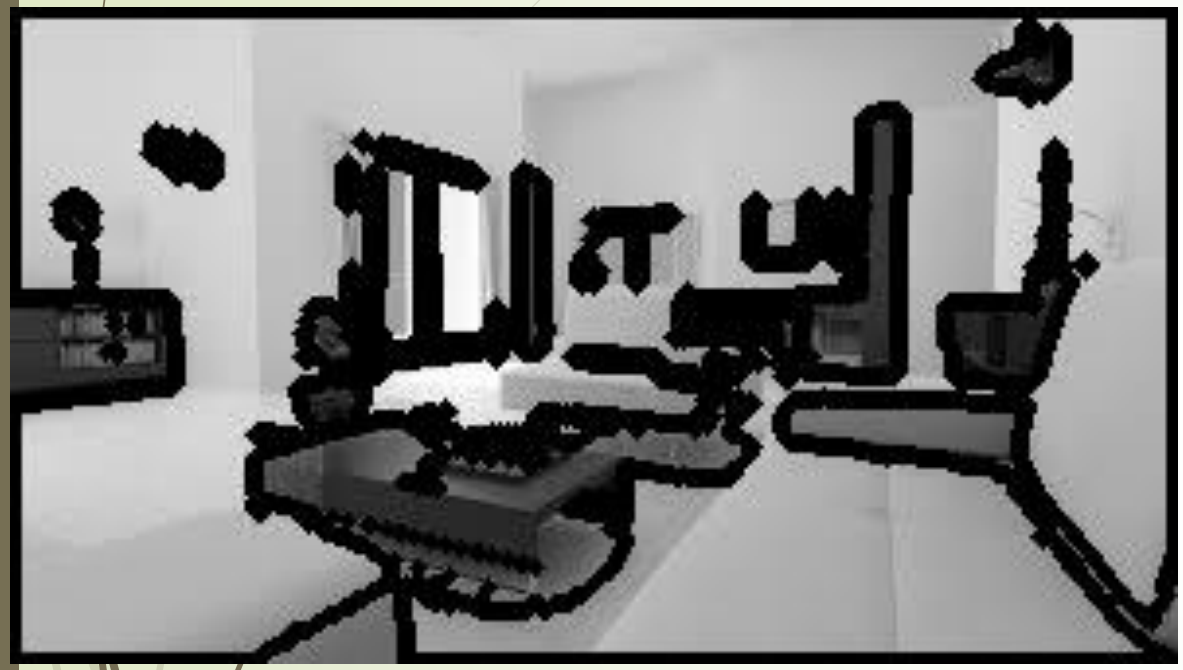
OpenCV

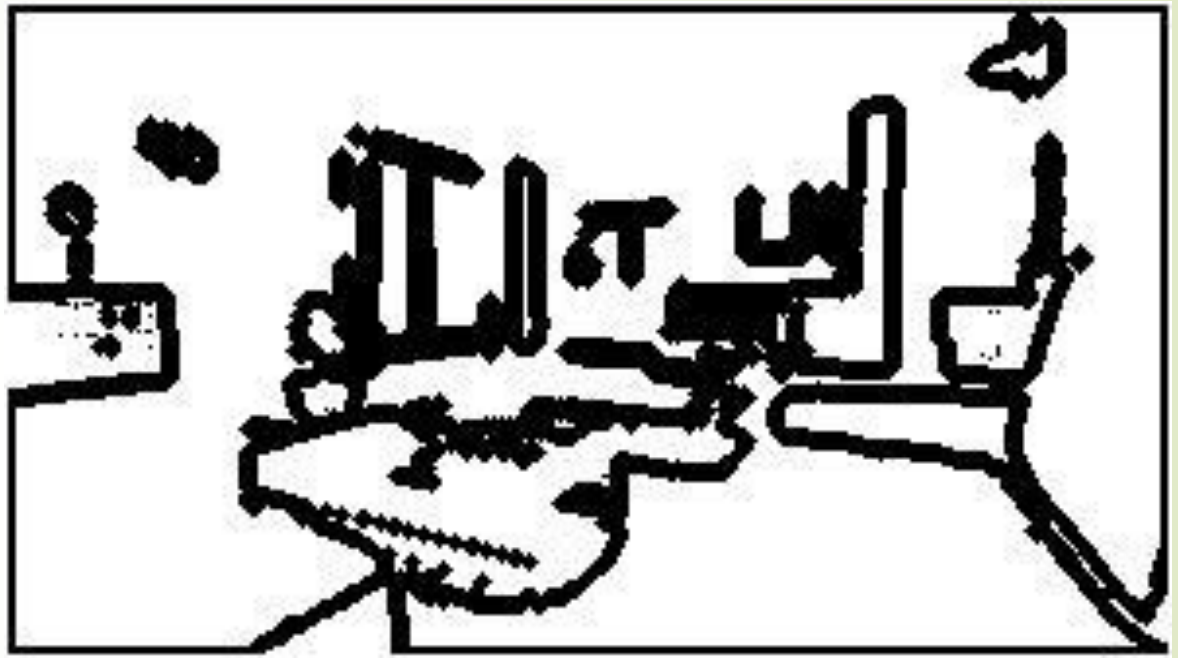

```
import cv2
from PIL import Image


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]))

```

Contours OpenCV



On enlève le bruit





Contours convolution

Contours OpenCV

Coefficient de contours

```
 7
 8 from PIL import Image
 9
10 def coef(URLImage):
11     image = Image.open(URLImage)
12     colonne,ligne = image.size
13     n=colonne*ligne
14     image_en_memoire = image.load()
15     somme=0
16     for x in range(colonne):
17         for y in range(ligne):
18             p = image_en_memoire[x,y]
19             if p==(255,255,255) :
20                 somme += 1
21     return(somme/n)
22
```

Traitement pour la banque d'image

- On supprime les images trop éloignés des caractéristiques
- On ajuste les couleurs moyennes des images

```
import os
monRepertoire='D:/Cours/MP/TIPE/Banque_image/salon'
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)
```

```
41 resolution = 0.1
42 if not os.path.exists('defectueuses'):
43     os.makedirs('defectueuses')
44
45
46 def modif_moyenne(URLImage,moyimage,moybanque):
47     image = Image.open(URLImage)
48     enMemoire = image.load()
49     colonne,ligne = image.size
50     r=int(moybanque[0]-moyimage[0])
51     v=int(moybanque[1]-moyimage[1])
52     b=int(moybanque[2]-moyimage[2])
53     p0=0
54     p1=0
55     p2=0
56     for i in range(colonne):
57         for j in range(ligne):
58             p=enMemoire[i,j]
59             p0 = p[0] + r
60             p1 = p[1] + v
61             p2 = p[2] + b
62             enMemoire[i,j]=(p0,p1,p2)
63     image.save('moyenne_modifie.'+URLImage)
64     image.close()
65
66 for image in fichiers:
67     if abs(coef(image)-coefficient)>resolution:
68         os.rename(image,monRepertoire+'/defectueuses/'+image)
69     else:
70         modif_moyenne(image,dic[image],somme)
71
```

(112, 111, 110)

(115, 116, 113)

(167, 164, 160)

(130, 129, 124)

(132, 130, 125)

(135, 133, 128)

Conclusion

- Un processus automatique permet de rendre une banque d'image homogène.
- Utile pour des sites commerçant, des marques.
- Idées pour la suite :
  - Caractéristiques d'une image :
    - Nombre de plans
    - Taille du premier plan
  - Application à la banque d'image :
    - Ajuster les moyenne de couleurs
    - Appliquer des filtres
    - Recadrage