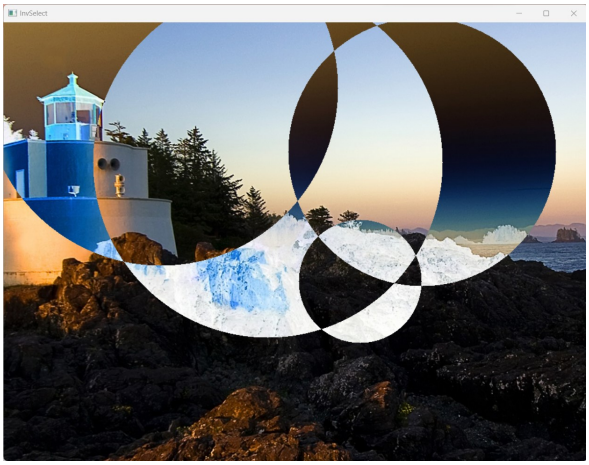
 # Inverser la zone sélectionnée (Les propriétés du disque sont définies à l'aide de la souris)

```
import cv2
import numpy as np

# Point Centre et point bord du cercle
params = {'x0':-1,
        'y0':-1,
        'x1':-1,
        'y1':-1,
        'Pressed':False
    }
img = cv2.imread('../images/a.jpg',1)

# Eliminer la composante rouge à l'intéreiur du disque
def traitement():
    tmpImg = np.zeros(img.shape,np.uint8)
    x = params['x1'] - params['x0']
    y = params['y1'] - params['y0']
    rayon = np.int16(np.linalg.norm([x,y]))
    cv2.circle(tmpImg,(params['x0'],params['y0']),rayon,(255,255,255),-1)

    img[tmpImg[:,:,2]==255,2] = 255-img[tmpImg[:,:,2]==255,2]
    img[tmpImg[:, :, 2] == 255, 0] = 255 - img[tmpImg[:, :, 2] == 255,0]
    img[tmpImg[:, :, 2] == 255, 1] = 255 - img[tmpImg[:, :, 2] == 255, 1]

# Déssiner le cercle
def dessinerCercle(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDOWN:
        params['x0'] = x
        params['y0'] = y
        params['Pressed'] = True
    elif event==cv2.EVENT_LBUTTONUP:
        params['Pressed'] = False
        traitement();
        cv2.imshow('InvSelect',img)
    elif event==cv2.EVENT_MOUSEMOVE and params['Pressed']:
        params['x1'] = x
        params['y1'] = y
        rayon = np.int(np.linalg.norm([params['x1'] - params['x0'],params['y1'] - params['y0']]))
        imgbis = img.copy()
        cv2.circle(imgbis,(params['x0'],params['y0']),rayon,(100,0,0),-1)
        cv2.imshow('InvSelect',imgbis)
```

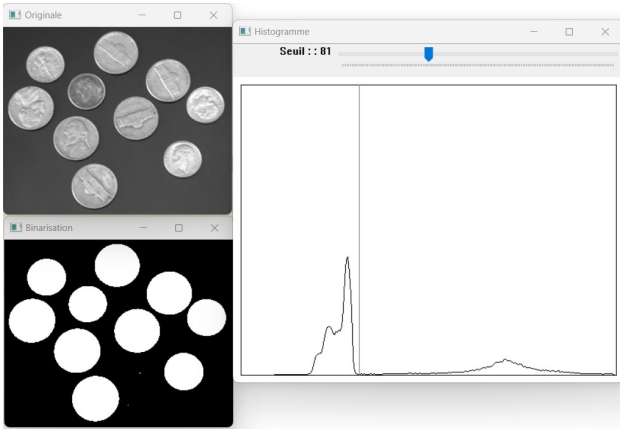
```
def main():
      cv2.namedWindow('InvSelect')
      cv2.imshow('InvSelect',img)
      # Gestion de la souris

cv2.setMouseCallback('InvSelect',dessinerCercle)
      while(True):
         if cv2.waitKey(20) & 0xFF == 27:
           break

      cv2.destroyAllWindows()

if __name__ == "__main__":
   main()
```



 # Binarisation avec visualisation du seuil et de l'histogramme

```
import numpy as np
import cv2

def f(x):
   pass

def main():

   a = cv2.imread("..\\images\\im11.jpg",0)
   cv2.imshow('Originale', a)
   # Binarisation
   seuil = 128
   b = np.uint8((a>=seuil)*255)

   # Ou encore
   # ret,b=cv2.threshold(a,seuil,255,cv2.THRESH_BINARY);

   # Calcul de l'histogramme
   histSize = 256
   hist = cv2.calcHist([a],[0],None,[histSize],[0,histSize-1])

   # calculer le graphe de l'histogramme et déssiner dans une image
   imHistW = 512
   imHistH = 400
   bord = 10
   binHistW =  (imHistW-2*bord)/histSize
   histImage = np.ones((imHistH, imHistW, 1), dtype=np.uint8)*255

   # Histogramme normalisé
```

```
    cv2.normalize(hist, hist)
    hist=hist*imHistH

    for i in range(1, histSize):
        cv2.line(histImage, ( int(bord+binHistW*(i-1)), int(imHistH-bord- hist[i-1]) ),
            ( int(bord+binHistW*i), int(imHistH-bord - hist[i])), 0)
    cv2.rectangle(histImage, ( bord,bord),(imHistW-bord, imHistH-bord ), 0)

    # Créer une autre image pour y ajouter la ligne correspondant au seuil
    histImagePlusLigne = histImage.copy()
    cv2.line(histImagePlusLigne, (int(bord+binHistW*seuil), imHistH-bord),
            ( int(bord+binHistW*seuil), bord), 128)

    cv2.imshow('Binarisation', b)
    cv2.imshow('Histogramme', histImagePlusLigne)

    cv2.createTrackbar('Seuil : ', 'Histogramme', 0, 255, f)
    cv2.setTrackbarPos('Seuil : ', 'Histogramme',seuil)

    while True:

        seuil = cv2.getTrackbarPos('Seuil : ', 'Histogramme')
        # Binarisation
        b = np.uint8((a>=seuil)*255)
        cv2.imshow('Binarisation', b)
        histImagePlusLigne = histImage.copy()
        cv2.line(histImagePlusLigne, (int(bord+binHistW*seuil), imHistH-bord),
            ( int(bord+binHistW*seuil), bord), 128)

        cv2.imshow('Histogramme', histImagePlusLigne)
        if cv2.waitKey(100) & 0xFF == 27 :
            break

    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```



 # Calculer et afficher l'histogramme d'une image à couleurs réelles

```
import cv2
from matplotlib import pyplot as plt

def main():
```

```
    # Lecture de l'image
    img = cv2.imread("..\\images\\a.jpg",1)
    cv2.imshow("Image originale", img)

    # Récupérer chaque canal séparement
    chans = cv2.split(img)
    colors = ("b", "g", "r")
    plt.figure()
    plt.title("Histogramme couleur")
    plt.xlabel("Bins")
    plt.ylabel("Nbre des Pixels")

    for (chan, color) in zip(chans, colors):
        # Créer l'histogramme du canal courant
        hist = cv2.calcHist([chan], [0], None, [256], [0,
256])
        plt.plot(hist, color=color)
        plt.xlim([0, 256])
    plt.show()

    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

 # Utiliser les couleurs dominantes pour le calcul de la similarité entre les images d'un dossier

```
import os
import cv2
import numpy as np
from sklearn.cluster import KMeans

def calculCouleursDominantes(image_path, num_colors=5):
    # Lire l'image
    img = cv2.imread(image_path)
    # Redimensionner
    img = cv2.resize(img, (100, 100))
    # Convertir dans Lab
    img_lab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)

    pixels = img_lab.reshape(-1, 3)     # Convertir en tableau de pixels

    # Kmeans
```

```
    kmeans = KMeans(n_clusters=num_colors, n_init=10,
random_state=100)
    kmeans.fit(pixels)

    # Couleurs dominantes
    couleursDominante = kmeans.cluster_centers_.astype(int)
    return couleursDominante

def calcul_dist(colors1, colors2):
    # Distance moyenne entre les couleurs dominantes
    sommedistances = 0
    for color1 in colors1:
        for color2 in colors2:
            distance = np.linalg.norm(color1 - color2)
            sommedistances += distance

    # Calculer la moyenne des distances
    moyenneDistances = sommedistances / (len(color1) ** 2)

    return moyenneDistances

# Chemin
dossier = './images/'

# Nombre des couleurs à calculer
nColors = 5

# Lecture
fichiers = os.listdir(dossier)

# Parcour et calcul des couleurs dominantes
colors= []
for fichier in fichiers:
    colors.append(calculCouleursDominantes(dossier+fichier,
nColors))

for i in range(len(fichiers)):
    for j in range(len(fichiers)):
        d = calcul_dist(colors[i], colors[j])
        print("Distance (", fichiers[i],", ",fichiers[j],") = ",f"{d:.2f}")
```

Sortie:

```
Distance ( obj16__245.png , obj16__245.png ) = 149.70
Distance ( obj16__245.png , obj16__70.png ) = 162.74
Distance ( obj16__245.png , obj1__40.png ) = 218.15
Distance ( obj16__245.png , obj1__95.png ) = 220.04
Distance ( obj16__245.png , obj27__125.png ) = 224.12
Distance ( obj16__245.png , obj27__190.png ) = 220.50
Distance ( obj16__70.png , obj16__245.png ) = 162.74
Distance ( obj16__70.png , obj16__70.png ) = 167.51
Distance ( obj16__70.png , obj1__40.png ) = 223.89
Distance ( obj16__70.png , obj1__95.png ) = 224.94
Distance ( obj16__70.png , obj27__125.png ) = 233.48
Distance ( obj16__70.png , obj27__190.png ) = 230.29
Distance ( obj1__40.png , obj16__245.png ) = 218.15
Distance ( obj1__40.png , obj16__70.png ) = 223.89
Distance ( obj1__40.png , obj1__40.png ) = 193.25
Distance ( obj1__40.png , obj1__95.png ) = 197.45
Distance ( obj1__40.png , obj27__125.png ) = 242.22
Distance ( obj1__40.png , obj27__190.png ) = 240.04
Distance ( obj1__95.png , obj16__245.png ) = 220.04
Distance ( obj1__95.png , obj16__70.png ) = 224.94
Distance ( obj1__95.png , obj1__40.png ) = 197.45
Distance ( obj1__95.png , obj1__95.png ) = 191.56
Distance ( obj1__95.png , obj27__125.png ) = 247.51
Distance ( obj1__95.png , obj27__190.png ) = 246.04
Distance ( obj27__125.png , obj16__245.png ) = 224.12
Distance ( obj27__125.png , obj16__70.png ) = 233.48
Distance ( obj27__125.png , obj1__40.png ) = 242.22
Distance ( obj27__125.png , obj1__95.png ) = 247.51
Distance ( obj27__125.png , obj27__125.png ) = 214.25
Distance ( obj27__125.png , obj27__190.png ) = 212.71
Distance ( obj27__190.png , obj16__245.png ) = 220.50
Distance ( obj27__190.png , obj16__70.png ) = 230.29
Distance ( obj27__190.png , obj1__40.png ) = 240.04
Distance ( obj27__190.png , obj1__95.png ) = 246.04
Distance ( obj27__190.png , obj27__125.png ) = 212.71
Distance ( obj27__190.png , obj27__190.png ) = 204.03
```


obj1__40.png


obj1__95.png


obj16__70.png


obj16__245.png


obj27__125.png


obj27__190.png