РОБОТА із ЗОБРАЖЕННЯМИ

Файл: Image_09_002

Детектори кутів. Алгоритм Харріс (Harris)

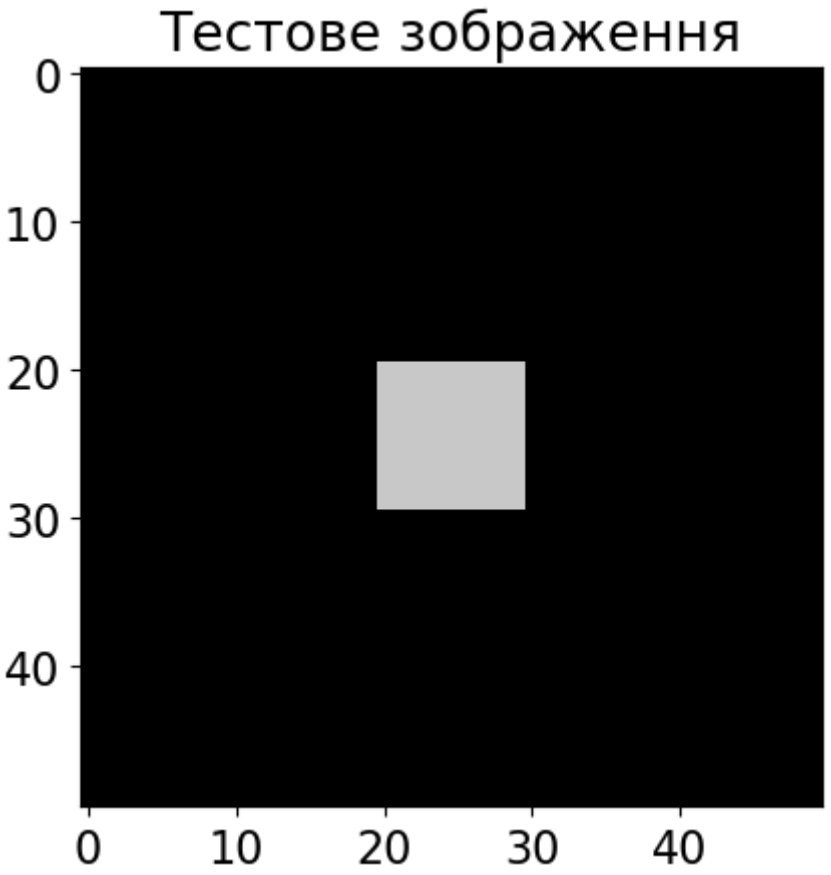
Підходи до побудови детекторів

```
## Завантаження пакетів
import numpy as np
import matplotlib.pyplot as plt
import skimage.io as io
from skimage.transform import resize
plt.rcParams['font.size'] = 16
from scipy import ndimage
```

Просте тестове зображення

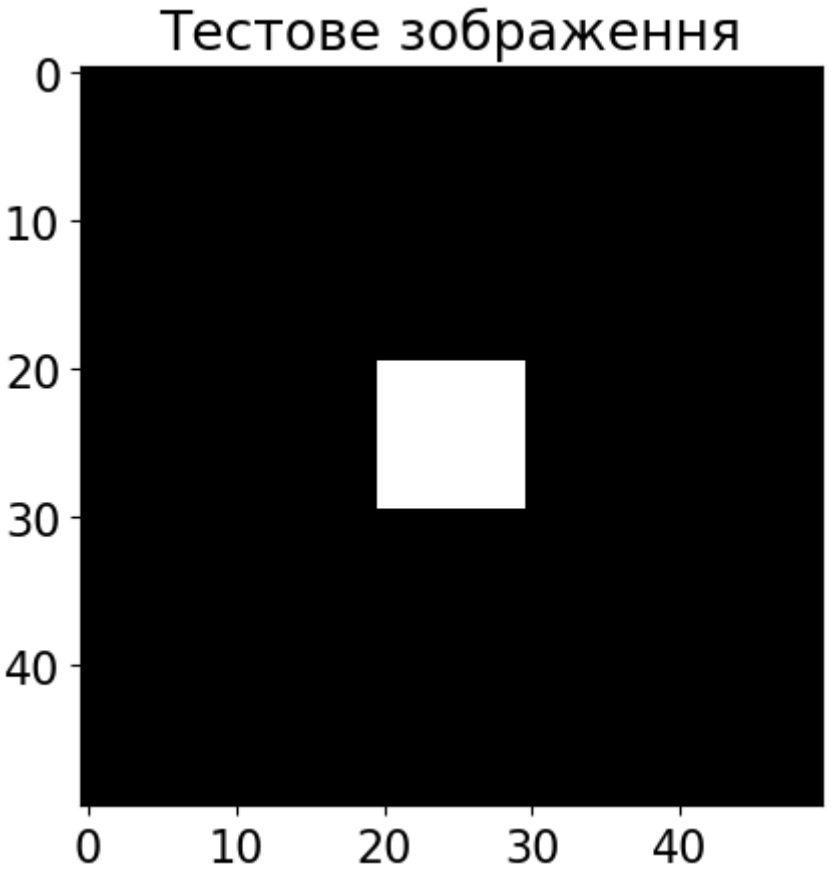
```
## Генерація зображення
test_img_simple = np.zeros ((50, 50, 3), dtype=np.uint8)

## Визначення стркутури та розміру зображення
print ('IMAGE SHAPE', test_img_simple.shape, 'IMAGE SIZE', test_img_simple.size)

irows_num = test_img_simple.shape[0] ## кількість рядків
iclms_num = test_img_simple.shape[1] ## кількість колонок
print ('ROWS NUMBER', irows_num, 'CLMS NUMBER', iclms_num)

for i in  range (20,30,1):
    for j in  range (20,30,1):
        test_img_simple[i, j, :] = [200,200,200]

plt.imshow(test_img_simple)
plt.title("Тестове зображення")
plt.show()
```

```
IMAGE SHAPE (50, 50, 3) IMAGE SIZE 7500
ROWS NUMBER 50 CLMS NUMBER 50
```



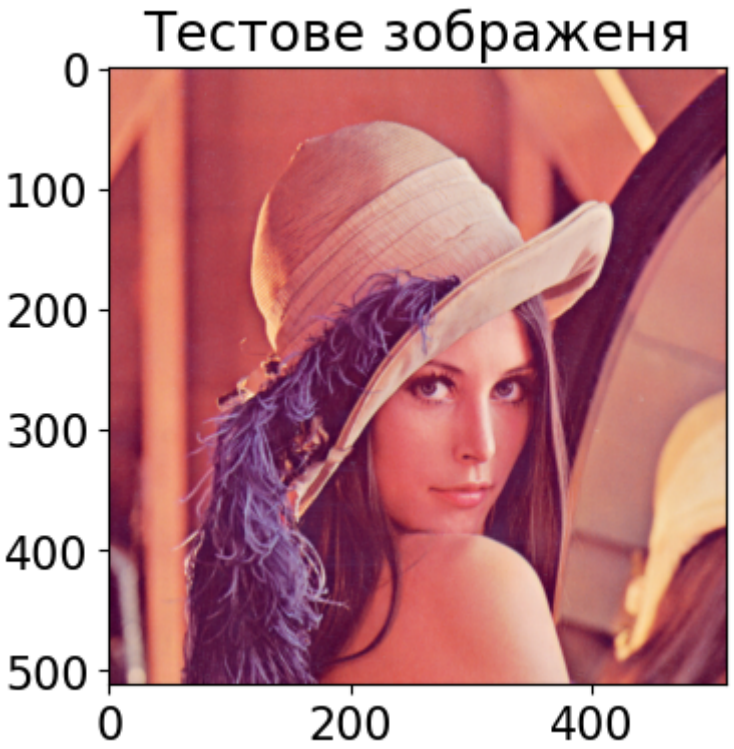
Використовуємо тільки одну кольорову компоненту

```
test_img_simple_ = np.zeros ((irows_num, iclms_num), dtype=np.uint8)
for i in  range (irows_num):
    for j in  range (iclms_num):
        test_img_simple_[i,j] =  test_img_simple[i,j,0]

plt.imshow(test_img_simple_, 'gray')
plt.title("Тестове зображення")
plt.show()
```




```
## Завантаження файла
path = './IMAGES/'
filenameP = 'Lenna.png'
# filenameP = 'Interest_Points_01.png'
#filenameP = 'Interest_Points_01_rotated.png'
# filenameP = 'Interest_Points_02.jpg'
test_img = io.imread(path + filenameP)

## Визначення стркутури та розміру зображення
print ('IMAGE SHAPE', test_img.shape, 'IMAGE SIZE', test_img.size)

irows_num = test_img .shape[0] ## кількість рядків
iclms_num = test_img .shape[1] ## кількість колонок
print ('ROWS NUMBER', irows_num, 'CLMS NUMBER', iclms_num)

fig, ax = plt.subplots(figsize=(4, 4))
plt.imshow(test_img )
plt.title("Тестове зображеня")
plt.show()
```

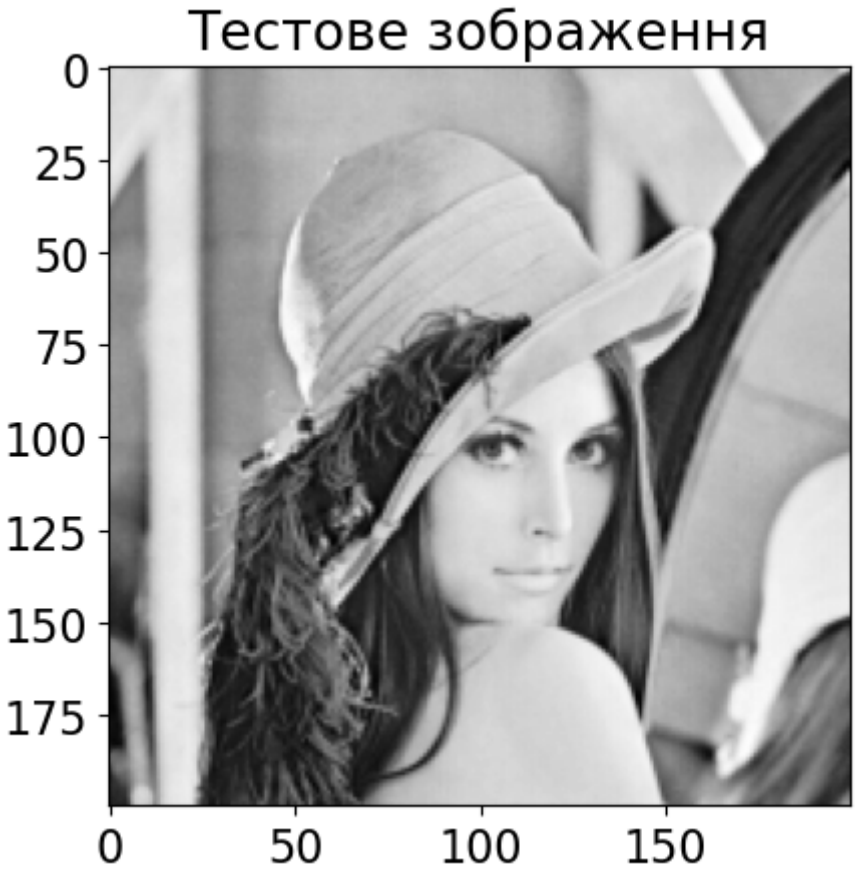
```
IMAGE SHAPE (512, 512, 3) IMAGE SIZE 786432
ROWS NUMBER 512 CLMS NUMBER 512
```




```
#Resize
irows_num_ = 200
iclms_num_ = 200
test_resize = resize(test_img, (irows_num_ , iclms_num_))
fig, ax = plt.subplots(figsize=(4, 4))
plt.imshow(test_resize )
plt.title("Тестове зображеня")
plt.show()
```

Використовуємо тільки одну кольорову компоненту

```
test_image = np.zeros ((irows_num_, iclms_num_), dtype=np.uint32)
for i in  range (irows_num_):
    for j in  range (iclms_num_):
        test_image[i,j] =  test_resize [i,j,0]*256

#print (test_resize[10,30])
#print (test_image[10,30])
plt.imshow(test_image, 'gray' )
plt.title("Тестове зображення")
plt.show()
```

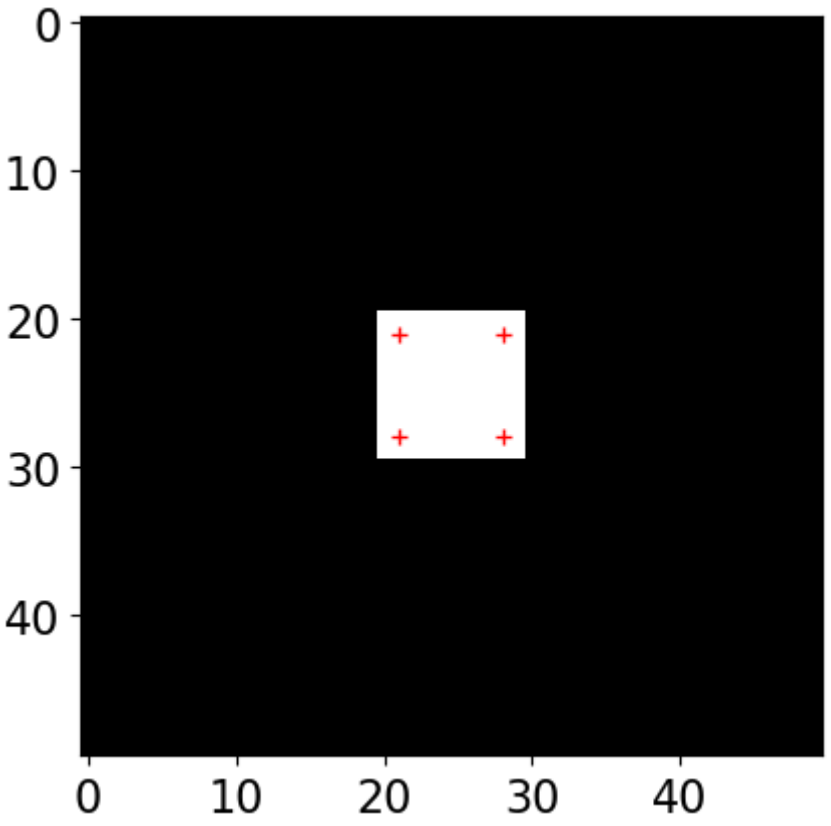


ФУНКЦИЯ ХАРРИСА

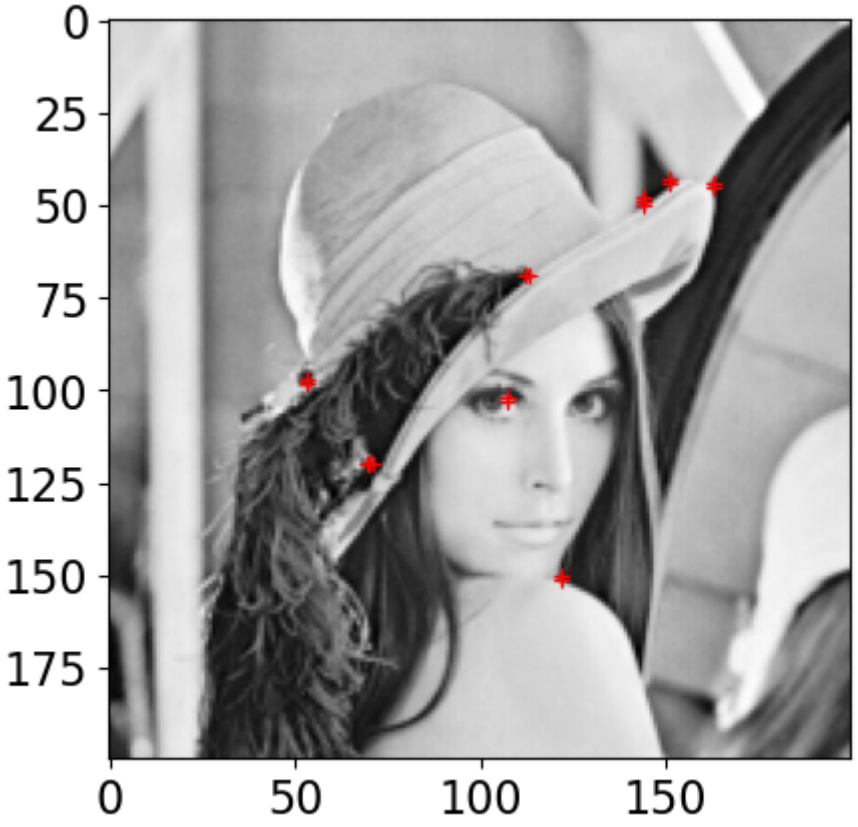
```
def Harris(im, threshold):
    # neim = imap1(im)
    neim = im
    imarr = np.asarray(neim, dtype=np.float64) # в массив с пл.зпт
    ix = ndimage.sobel(imarr, 0) # градиент по X , оператор Собеля
    iy = ndimage.sobel(imarr, 1) # градиент по H , оператор Собеля
    ix2 = ix * ix
    iy2 = iy * iy
    ixy = ix * iy
    # Усереднюємо (Гаус)
    ix2 = ndimage.gaussian_filter(ix2, sigma=2)
    iy2 = ndimage.gaussian_filter(iy2, sigma=2)
    ixy = ndimage.gaussian_filter(ixy, sigma=2)
    c, l = imarr.shape
    result = np.zeros((c, l))
    r = np.zeros((c, l))
    rmax = 0
    for i in range(c):
        # print('loking for corner . . .')
        for j in range(l):
            # print('test ',j)
            m = np.array([[ix2[i, j], ixy[i, j]], [ixy[i, j], iy2[i, j]]],
dtype=np.float64)
            # Обчислюємо відклик Хариса для кожного пікселя
            r[i, j] = np.linalg.det(m) - 0.04 * (np.power(np.trace(m), 2))
            # Визначаємо МАКСИМАЛЬНИЙ відклик
```

```
            if r[i, j] > rmax:
                rmax = r[i, j]
    for i in range(c - 1):
        # аналіз по рядкам
        for j in range(l - 1):
            # аналіз по колонкам
            #  > 0.002 это частка макимадльного відклика, що буде враховувати пікселі
            if r[i, j] > threshold * rmax and r[i, j] > r[i-1, j-1] and r[i, j] > r[i-1, j+1]\
                                    and r[i, j] > r[i+1, j-1] and r[i, j] > r[i+1, j+1]:
                result[i, j] = 1

    pc, pr = np.where(result == 1)
    plt.plot(pr, pc, 'r+')
    plt.savefig('harris_test.png')
    plt.imshow(im, 'gray')
    plt.show()
```

```
Harris(test_img_simple_,0.5)
```




```
Harris(test_image,0.4)
```


0
25
50
75
100
125
150
175
0
50
100
150