РОБОТА із ЗОБРАЖЕННЯМИ

Файл: Image_8_001

Метрики порівняння зображень

Примітка. В наступних прикладах зображення, що порівнюються, мають однаковий розмір

```
%matplotlib inline
```

```
import numpy as np
import matplotlib.pyplot as plt
import skimage.io as io
from skimage.color import rgb2gray
from scipy import ndimage
plt.rcParams['font.size'] = 10
```

ВИЗНАЧЕННЯ МЕТРИК

Функції метрик приймают два параметри: два ахроматичних зображення однакового шейпу

- im1 : TYPE numpy array.
  DESCRIPTION: gray image
- im2 : TYPE numpy array.
  DESCRIPTION: gray image

Функції повертають

- res: TYPE float. DESCRIPTION: обчислена метрика

```
# RSSD Сума квадратів різниць
def ssd(im1, im2):
    r = im1.shape[0]
    c = im1.shape[1]

    s = 0.0
    for i in range(r):
        for j in range(c):
            s += (im1[i, j]-im2[i, j])*(im1[i, j]-im2[i, j])
    s = 1.0 - s/(r*c)
    return s
```

```python
# Середньоквадратична похибка
def sqerror(im1, im2):
    r = im1.shape[0]
    c = im1.shape[1]
    s = 0.0
    for i in range(r):
        for j in range(c):
            s += (im1[i, j]-im2[i, j])*(im1[i, j]-im2[i, j])
    s = 1.0 - np.sqrt(s/(r*c))
    return s
```

```python
# Нормована кореляція
def cor(im1, im2):
    r = im1.shape[0]
    c = im1.shape[1]

    res = 0.000001
    im1_sqr = 0.000001
    im2_sqr = 0.000001
    for i in range(r):
        for j in range(c):
            res += im1[i, j]*im2[i, j]
            im1_sqr += im1[i, j]*im1[i, j]
            im2_sqr += im2[i, j]*im2[i, j]
    #print('--->', res, im1_sqr, im2_sqr)
    res = res / (np.sqrt(im1_sqr) * np.sqrt(im2_sqr))
    return res
```

```python
# Метрика Мінковського
def mink(im1, im2):
    r = im1.shape[0]
    c = im1.shape[1]

    res = 0.0

    for i in range(r):
        for j in range(c):
            res += np.abs(im1[i, j]-im2[i, j])
    res = 1.0 - res / (r * c)
    return res
```

```python
# Метрика Хаусдорфа
def hausdorf(im1, im2):

    r = im1.shape[0]
    c = im1.shape[1]

    res = 0.0

    for i in range(r):
        for j in range(c):
            dif = np.abs(im1[i, j]-im2[i, j])
```

```
            if dif > res:
                res = dif
    res = 1.0 - res
    return res
```

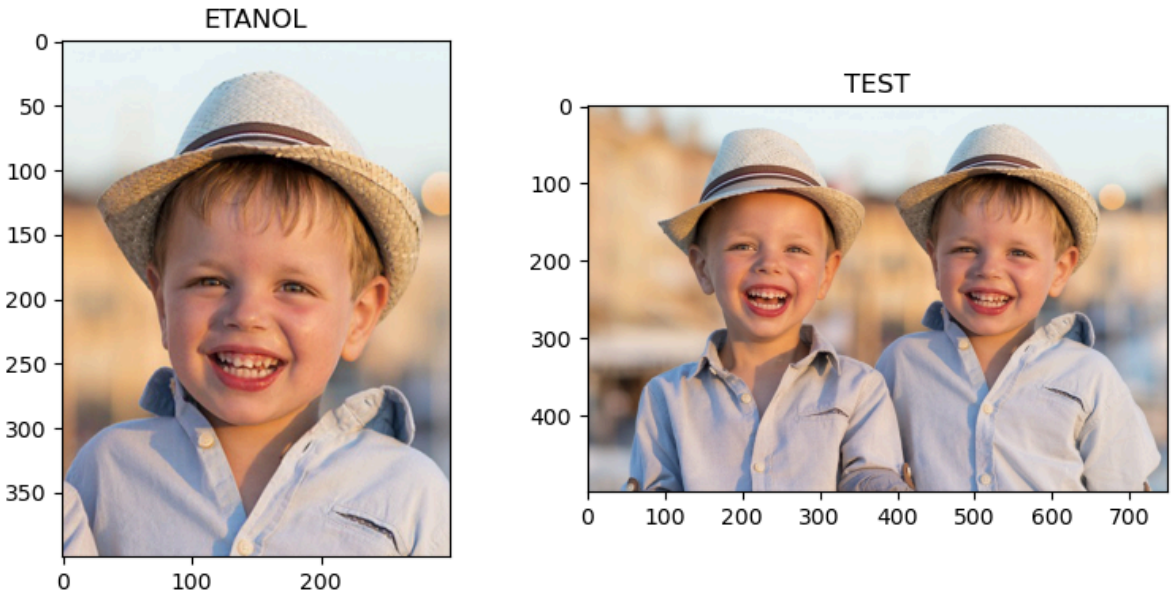
```
# Завантаження еталонного зображення
path = './IMAGES/'
# filename1 = 'Face_1_300_X_400.jpg'
filename1 = 'Face_2_300_X_400.jpg'
# filename1 = 'Face_3_300_X_400.jpg'
etalon_img = io.imread(path+filename1)
#  Визначення стркутури та розміру зображення
print('---- ETALON -----')
print('IMAGE SHAPE', etalon_img.shape, 'IMAGE SIZE', etalon_img .size)
rows_etalon = etalon_img.shape[0]  # кількість рядків етанолног зображення
clms_etalon = etalon_img.shape[1]  # кількість колонок етанолног зображення
print('ROWS NUMBER', rows_etalon, 'CLMS NUMBER', clms_etalon)
```

```
---- ETALON -----
IMAGE SHAPE (400, 300, 3) IMAGE SIZE 360000
ROWS NUMBER 400 CLMS NUMBER 300
```

```
# Завантаження зображення, що обробляється
path = './IMAGES/'
filename2 = 'Faces_two_.jpg'
# filename2 = 'Faces_many_.jpg'

test_img = io.imread(path+filename2)

#  Визначення стркутури та розміру зображення
print('---- ЗОБРАЖЕННЯ ЩО ОБРОБЛЯЄТЬСЯ -----')
print('IMAGE SHAPE', test_img .shape, 'IMAGE SIZE', test_img .size)
rows_test = test_img.shape[0]  # кількість рядків
clms_test = test_img.shape[1]  # кількість колонок
print('ROWS NUMBER', rows_test, 'CLMS NUMBER', rows_test)
```

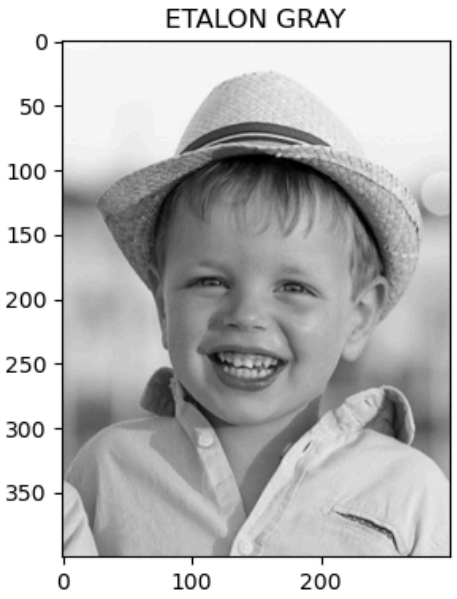
```
---- ЗОБРАЖЕННЯ ЩО ОБРОБЛЯЄТЬСЯ -----
IMAGE SHAPE (499, 750, 3) IMAGE SIZE 1122750
ROWS NUMBER 499 CLMS NUMBER 499
```

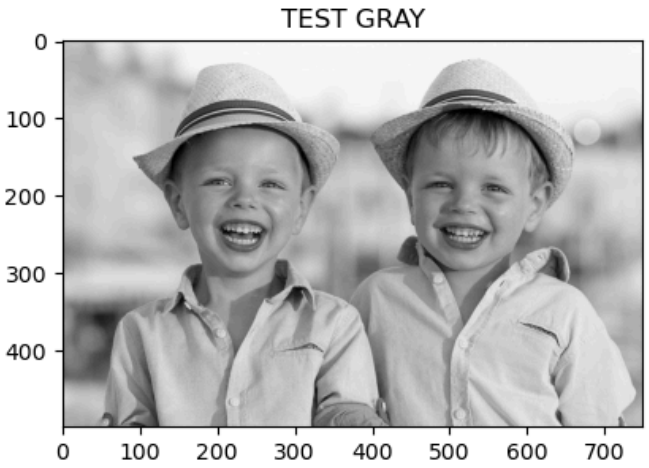
```
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(etalon_img)
ax[0].set_title("ETANOL")
ax[1].imshow(test_img)
ax[1].set_title("TEST")

fig.tight_layout()
plt.show()
```




```
# Перетворення до сірого
etalon_img_gray = rgb2gray(etalon_img)
test_img_gray = rgb2gray(test_img)
```

```
fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(etalon_img_gray,  cmap=plt.cm.gray )
ax[0].set_title("ETALON GRAY")
ax[1].imshow(test_img_gray,  cmap=plt.cm.gray)
ax[1].set_title("TEST GRAY")

fig.tight_layout()
plt.show()
```





Визначення зсувів

```
step_rows = 15  # Крок по ветикалі
step_clms = 20  # Крок по горізонталі

rows_diff = rows_test - rows_etalon
clms_diff = clms_test - clms_etalon

rows_steps = rows_diff // step_rows  # Кількуість зсувів по вертикалі
clms_steps = clms_diff // step_clms  #  Кількуість зсувів по горизонтали

print('РІЗНИЦЯ ПО ВЕРТИКАЛІ   ', rows_diff, 'КРОКІВ ', rows_steps)
print('РІЗНИЦЯ ПО ГОРИЗОНТАЛІ ', clms_diff, 'КРОКІВ', clms_steps)
```

```
РІЗНИЦЯ ПО ВЕРТИКАЛІ    99 КРОКІВ  6
РІЗНИЦЯ ПО ГОРИЗОНТАЛІ  450 КРОКІВ 22
```

КОВАЗЕНЕ ВІКНО

```
# Поточний фрагмент тестовго зображення "під" еталоном
fragment = np.zeros((rows_etalon, clms_etalon), dtype=np.float32)
# Значення метрики для поточного положення еталону
metrics_array = np.zeros((rows_steps, clms_steps), dtype=np.float32)

for row in range(rows_steps):
    for clm in range(clms_steps):

        # print('Позиція етлону', row, clm)
        fragment[:, :] = test_img_gray[row*step_rows:row*step_rows+rows_etalon,
                                       clm*step_clms:clm*step_clms+clms_etalon]
        # Взначаємо метрику
        metrics_array[row, clm] = hausdorf(etalon_img_gray, fragment)
        # hsdrf_array[row, clm] = mink(etalon_img_gray, fragment)
```

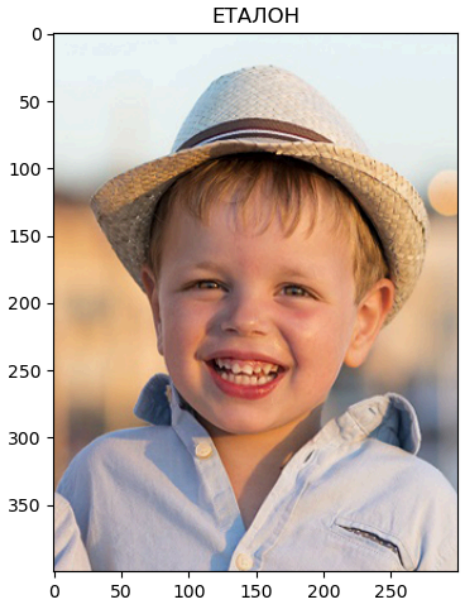
```
        # hsdrf_array[row, clm] = cor(etalon_img_gray, fragment)
```

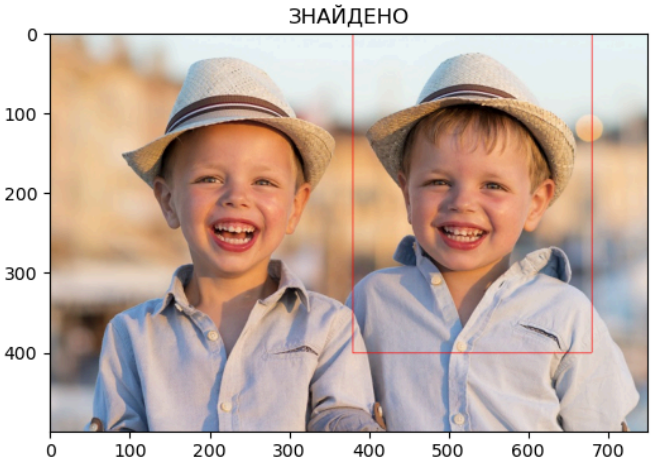
```
# Координти куів  ВОХу
print('-------------------------------------')
max_index = np.unravel_index(np.argmax(metrics_array), metrics_array.shape)
max_value = metrics_array[max_index]
print('MAX', max_index, max_value)

frame_corners = np.zeros((4, 2), dtype=np.uint16)
frame_corners[0, 0] = max_index[0] * step_rows
frame_corners[1, 0] = max_index[0] * step_rows
frame_corners[2, 0] = max_index[0] * step_rows + rows_etalon
frame_corners[3, 0] = max_index[0] * step_rows+rows_etalon

frame_corners[0, 1] = max_index[1] * step_clms
frame_corners[1, 1] = max_index[1] * step_clms + clms_etalon
frame_corners[2, 1] = max_index[1] * step_clms
frame_corners[3, 1] = max_index[1] * step_clms + clms_etalon

for i in range(4):
    print('Кут', i, 'Координти', frame_corners[i, 0], frame_corners[i, 1])
```

```
-------------------------------------
MAX (0, 19) 0.21140781
Кут 0 Координти 0 380
Кут 1 Координти 0 680
Кут 2 Координти 400 380
Кут 3 Координти 400 680
```

ЗОБРАЖЕННЯ

```
# Побудова боксу на тестовому зображені
for j in range(max_index[1] * step_clms, max_index[1] *
               step_clms + clms_etalon):
    test_img[max_index[0] * step_rows, j, :] = [255, 0, 0]
    test_img[max_index[0] * step_rows + rows_etalon, j, :] = [255, 0, 0]
for i in range(max_index[0] * step_rows, max_index[0] *
               step_rows + rows_etalon):
    test_img[i, max_index[1] * step_clms, :] = [255, 0, 0]
    test_img[i, max_index[1] * step_clms + clms_etalon, :] = [255, 0, 0]


fig, axes = plt.subplots(1, 2, figsize=(10, 5))
ax = axes.ravel()
ax[0].imshow(etalon_img)
ax[0].set_title("ЕТАЛОН")
ax[1].imshow(test_img)
ax[1].set_title("ЗНАЙДЕНО")
fig.tight_layout()
plt.show()
```


ЕТАЛОН
0
50
100
150
200
250
300
350
0
50
100
150
200
250


ЗНАЙДЕНО
0
100
200
300
400
0
100
200
300
400
500
600
700