# РОБОТА із ЗОБРАЖЕННЯМИ

## Файл: Image_8_003

## Порівняння зображень. Перцептивний хеш. Метрика Хемінга

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 skimage.transform import rescale, resize, downscale_local_mean

plt.rcParams['font.size'] = 10
```

## Відстань Хемінга (між хешами)

Функції метрик приймают два параметри: два ахроматичних зображення однакового шейпу

- hash1 : TYPE list of 256 ints (0 or 1)
  DESCRIPTION: хеш першого зображення
- hash2 : TYPE list of 256 ints (0 or 1)
  DESCRIPTION: хеш другого зображення
  Функції повертають
- res: TYPE int. DESCRIPTION: кількість різних бітів

```
def distance(hash1, hash2):
    len_hash1 = len(hash1)
    len_hash2 = len(hash2)
    if len_hash1 == len_hash2:  # перівірка рівності довжини кодів
        result = 0
        for i in range(len_hash1):
            if hash1[i] != hash2[i]:
                result += 1
        return result

    else:   # Довжина хешів не однакова
        return 'Number of bits is not equal'
```

```
# Simple Test
binary_code_1 = [1,1,1,1,1,1,1,1,1]
binary_code_2 = [0,0,0,0,0,0,0,1,0]
print (binary_code_1)
print (binary_code_2)
print (distance(binary_code_1,binary_code_2))
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 1, 0]
8
```

Хеш обчислюємо для зображень розміром 16 X 16 пікселів.
Тобто

1. зображення перетворюїмо до розміру 16 X 16
2. перетворюмо до сірого
3. обчислюємо середнє значення яскравості
4. в хеш звавантажуємо 0 для пікселя що менш середнього, 1 - для пікселя більш середнього

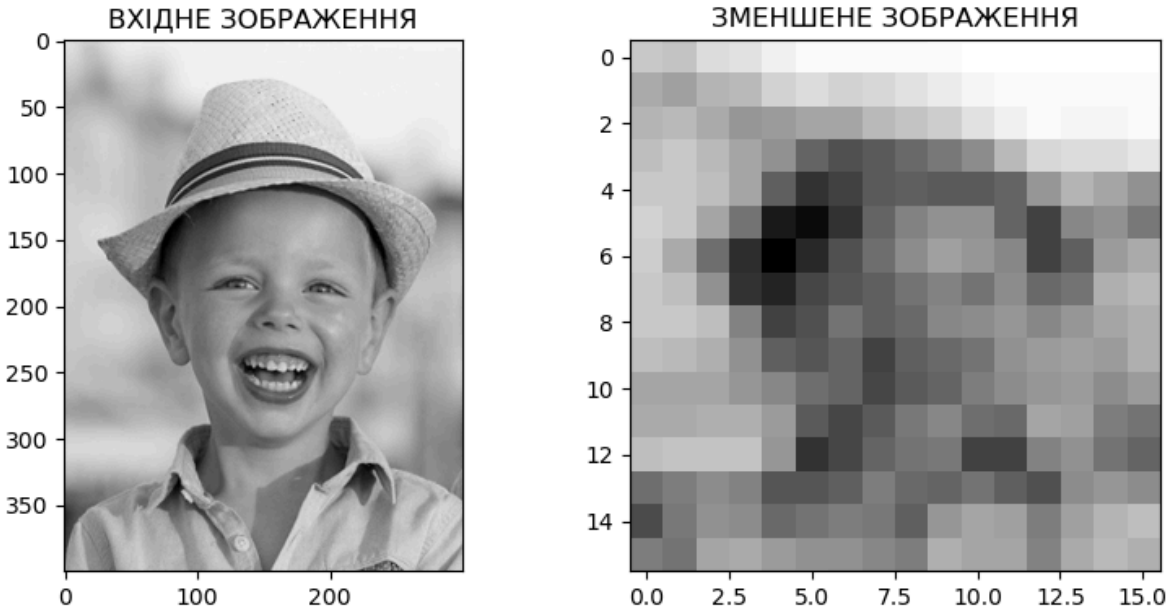
```
image_size = 16
```

```
# Завантаження першого зображення
path = './IMAGES/'
filename1 = 'Face_1_300_X_400.jpg'
# filename1 = 'Face_2_300_X_400.jpg'
# filename1 = 'Face_3_300_X_400.jpg'
img_1 = io.imread(path+filename1)
#  Визначення стркутури та розміру зображення
print('---- ЗОБРАЖЕННЯ 1 -----')
print('IMAGE SHAPE', img_1.shape, 'IMAGE SIZE', img_1.size)
rows_img_1 = img_1.shape[0]  # кількість рядків
clms_img_1 = img_1.shape[1]  # кількість колонок
print('ROWS NUMBER', rows_img_1, 'CLMS NUMBER', clms_img_1)
```

```
---- ЗОБРАЖЕННЯ 1 -----
IMAGE SHAPE (400, 300, 3) IMAGE SIZE 360000
ROWS NUMBER 400 CLMS NUMBER 300
```

```
# Формуємо зменшене ахроматичне зображення
img_1_gray = rgb2gray(img_1)
img_1_resized = resize(
    img_1_gray, (image_size, image_size), anti_aliasing=True)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(img_1_gray, cmap=plt.cm.gray)
ax[0].set_title("ПЕРШЕ ВХІДНЕ ЗОБРАЖЕННЯ")
ax[1].imshow(img_1_resized, cmap=plt.cm.gray)
ax[1].set_title("ЗМЕНШЕНЕ ЗОБРАЖЕННЯ")
```

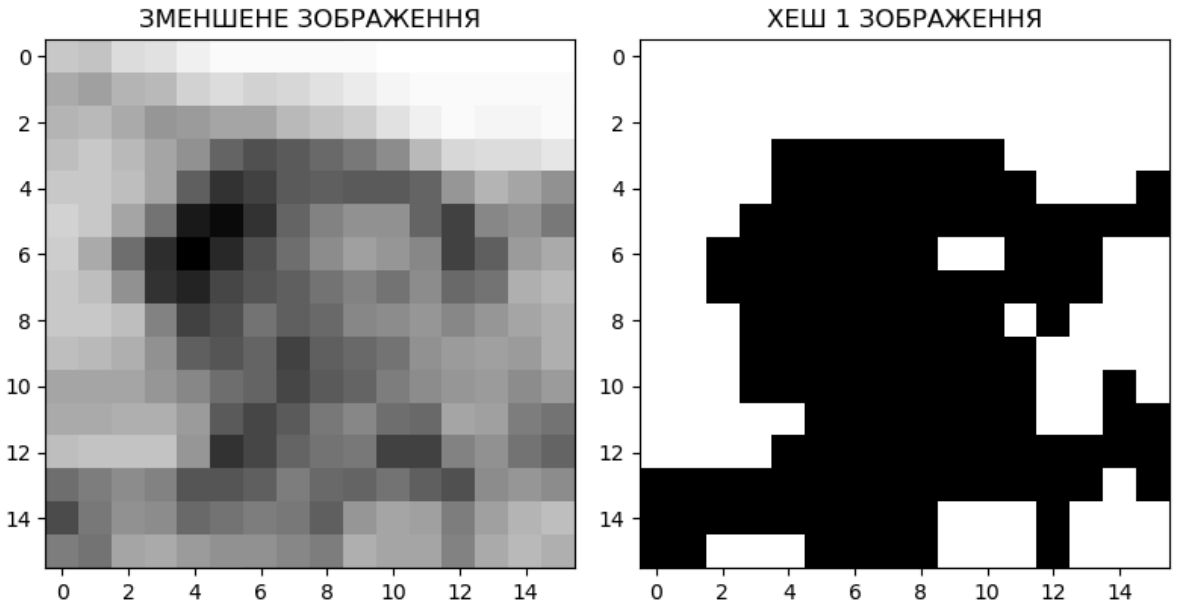
```
fig.tight_layout()
plt.show()
```




```
# Формуємо хеш першого зображення
aver_1_ = np.average(img_1_resized)
print('AVERAGE IMGE 1 --> ', aver_1_)

hash_1 = [0] * (image_size * image_size)
hash_1_image = np.zeros((image_size, image_size), dtype=np.float32)
for i in range(image_size):
    for j in range(image_size):
        if img_1_resized[i, j] > aver_1_:
            hash_1[i*image_size + j] = 1
            hash_1_image[i, j] = 1.0

#print (hash_1)
print (hash_1_image)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(img_1_resized, cmap=plt.cm.gray)
ax[0].set_title("ЗМЕНШЕНЕ ЗОБРАЖЕННЯ")
ax[1].imshow(hash_1_image, cmap=plt.cm.gray)
ax[1].set_title("ХЕШ 1 ЗОБРАЖЕННЯ")
fig.tight_layout()
plt.show()
```

```
AVERAGE IMGE 1 -->  0.6550731159489114
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

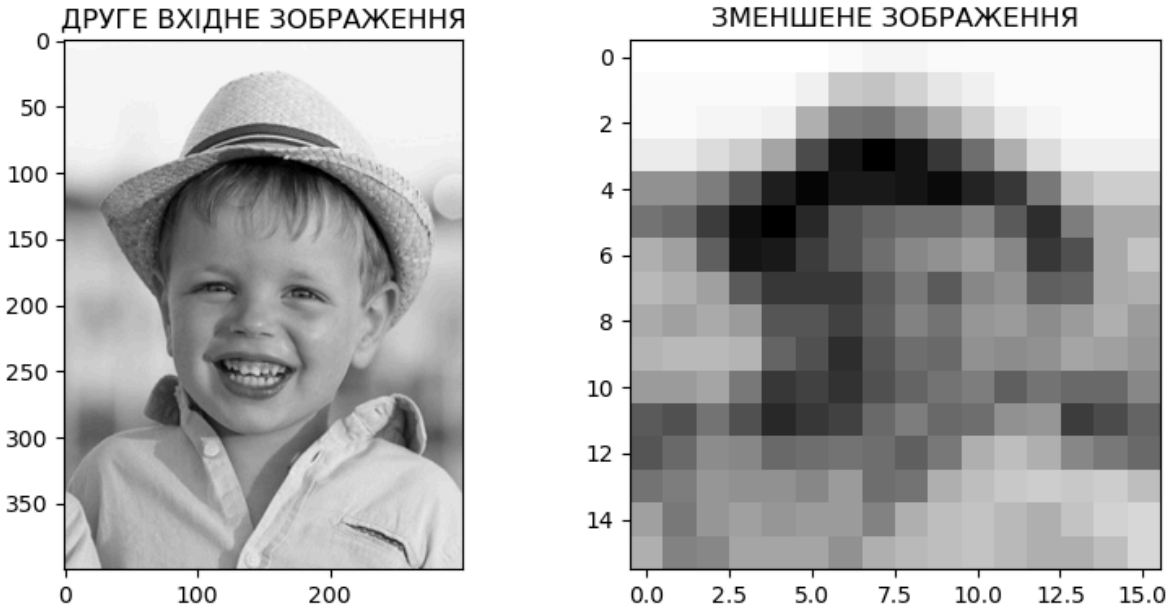
```
 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 1.]
 [1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1.]
 [1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1.]
 [0. 0. 1. 1. 1. 0. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1.]]
```




```python
# Завантаження другого зображення
path = './IMAGES/'
# filename2 = 'Face_1_300_X_400.jpg'
filename2 = 'Face_2_300_X_400.jpg'
# filename2 = 'Face_1_300_X_400_180.jpg'
# filename2 = 'Face_1_300_X_400_090.jpg'
# filename2 = 'Face_1_300_X_400_180.jpg'
# filename2 = 'Face_1_300_X_400_270.jpg'
# filename2 = 'Faces_many_.jpg'
img_2 = io.imread(path+filename2)
#  Визначення стркутури та розміру зображення
print('---- ЗОБРАЖЕННЯ 2 -----')
print('IMAGE SHAPE', img_2.shape, 'IMAGE SIZE', img_2.size)
rows_img_2 = img_2.shape[0]  # кількість рядків
clms_img_2 = img_2.shape[1]  # кількість колонок
print('ROWS NUMBER', rows_img_2, 'CLMS NUMBER', clms_img_2)
```

```
---- ЗОБРАЖЕННЯ 2 -----
IMAGE SHAPE (400, 300, 3) IMAGE SIZE 360000
ROWS NUMBER 400 CLMS NUMBER 300
```

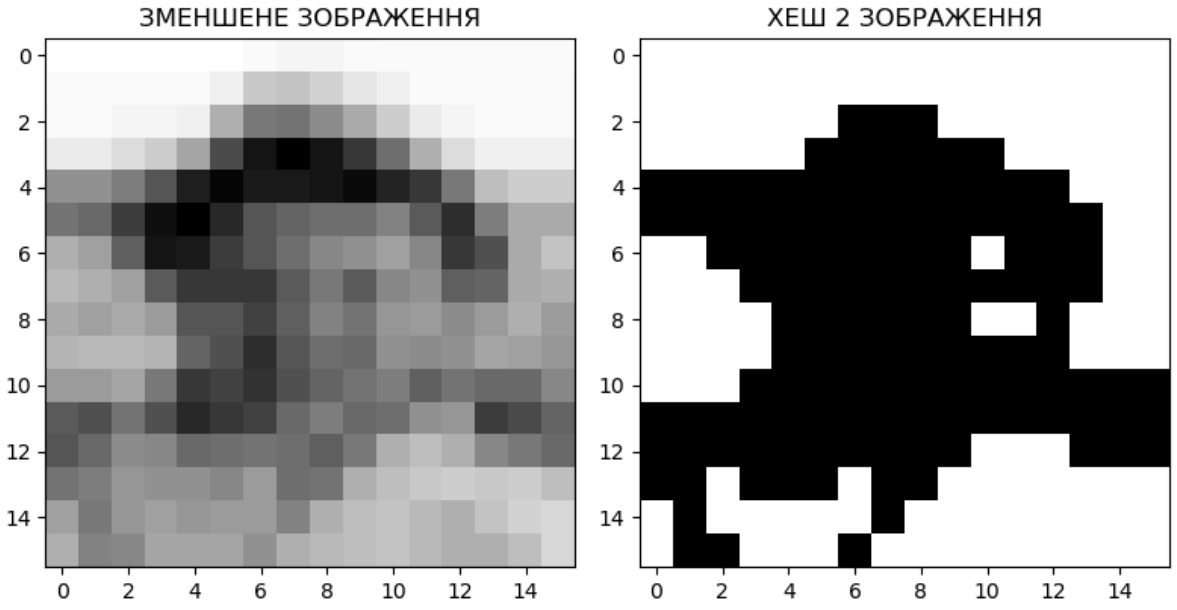
```python
# Формуємо зменшене ахроматичне зображення
img_2_gray = rgb2gray(img_2)
img_2_resized = resize(
    img_2_gray, (image_size, image_size), anti_aliasing=True)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(img_2_gray, cmap=plt.cm.gray)
ax[0].set_title("ДРУГЕ ВХІДНЕ ЗОБРАЖЕННЯ")
ax[1].imshow(img_2_resized, cmap=plt.cm.gray)
ax[1].set_title("ЗМЕНШЕНЕ ЗОБРАЖЕННЯ")
fig.tight_layout()
plt.show()
```




```python
# Формуємо хеш першого зображення
aver_2_ = np.average(img_2_resized)
print('AVERAGE IMGE 2 --> ', aver_2_)

hash_2 = [0] * (image_size * image_size)
hash_2_image = np.zeros((image_size, image_size), dtype=np.float32)
for i in range(image_size):
    for j in range(image_size):
        if img_2_resized[i, j] > aver_2_:
            hash_2[i*image_size + j] = 1
            hash_2_image[i, j] = 1.0
```

```
#print (hash_2)
print (hash_2_image)

fig, axes = plt.subplots(1, 2, figsize=(8, 4))
ax = axes.ravel()
ax[0].imshow(img_2_resized, cmap=plt.cm.gray)
ax[0].set_title("ЗМЕНШЕНЕ ЗОБРАЖЕННЯ")
ax[1].imshow(hash_2_image, cmap=plt.cm.gray)
ax[1].set_title("ХЕШ 2 ЗОБРАЖЕННЯ")
fig.tight_layout()
plt.show()
```

```
AVERAGE IMGE 2 -->  0.6668580833073994
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 1.]
 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.]
 [1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```



## Обчислюємо відстань Хаусдорфа

```
print('---------------------------------------')
print('Відстань Хаусдорфа = ', distance(hash_1, hash_2))
print('---------------------------------------')
```

```
---------------------------------------
Відстань Хаусдорфа =  60
---------------------------------------
```

```

```