Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра Интеллектуальных Технологий в Гуманитарной Сфере

Отчет по лабораторной работе №1

Факультет ФИКТ

Группа №K3242

Пыхтин Михаил

Преподаватель:

Хлопотов Максим Валерьевич

Исходное изображение

Импортируем библиотеки

In [1]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
from numpy import clip
from skimage import img_as_float, img_as_ubyte, io
from skimage.io import imread, imshow, imsave
from skimage.color import rgb2yuv, yuv2rgb, rgb2gray
```

Чтение исходной картинки

In [68]:
```
img = imread('13.jpg')
image = img_as_float(img)

imshow(img)
```

Out[68]:
```
<matplotlib.image.AxesImage at 0x910448e940>
```

Сразу запишем функции нахождения энтропии и среднеквадратической ошибки(mse)

In [47]:
```
def entropy(img):
    frcy = np.array([0 for i in range(256)])
    for row in img:
        for px in row:
            frcy[px] += 1

    n = len(img) * len(img[0])
    frcy = frcy / n
    ent = -np.sum([p * np.log2(p) for p in frcy if p != 0])
    return ent
def mse(img1, img2):
    tmp = np.sum([(px1 - px2) ** 2 for px1, px2 in zip(img1[:,:], img2[:,:])])
    mse = np.sqrt(tmp / len(img2) / len(img2[0]))
    return mse
```

Перевод картинки в greyscale

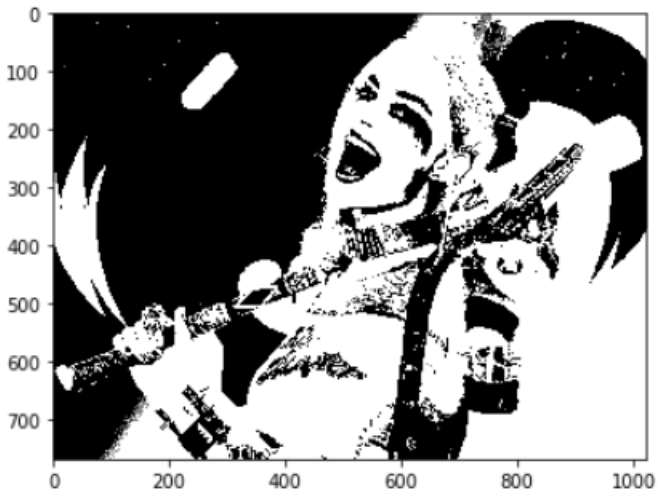
```
In [69]: gray = img[:, :,0]
         io.imshow(gray)
         io.imsave('gray.png', gray)
         entropy(gray)
```

```
Out[69]: 7.5857181130279407
```

Уменьшаем количество уровней яркости(Таким же способом можно сделать любое кол-во уровней, 2 в нашем случае)

```
In [70]: low = img_as_ubyte(gray)
         for i in low:
             for l in range(len(i)):
                 if 127 > i[l] > 0:
                     i[l] = 0
                 elif 255 > i[l] > 128:
                     i[l] = 255
         io.imsave('grayl.jpg', low)
         io.imshow(low)
         entropy(low)
```

```
Out[70]: 1.0619118254326652
```



Посчитаем энтропию исходного изображения

```
entropy(img)
```

```
Out[77]: 18.349437696477732
```

Теперь переведем исходное изображение в формат YUV и начинаем прореживать каналы U V, с разбиением на фрагменты 3x3

```
In [78]: img_yuv = rgb2yuv(image)
```

```
In [79]: def decim(img, k):
             img = np.array(img, dtype="float")
             y, u, v = img[..., 0], img[..., 1], img[..., 2]
             new_u = np.array([np.array([px for j, px in enumerate(row) if j % 2])
                 for i, row in enumerate(u) if i % 2])
             new_v = np.array([np.array([px for j, px in enumerate(row) if j % 2])
                 for i, row in enumerate(v) if i % 2])
             new_u = np.array(np.repeat([np.repeat([px for px in row], 2) for row in new_u], 2, axis=0))
             new_v = np.array(np.repeat([np.repeat([px for px in row], 2) for row in new_v], 2, axis=0))
             new_img = np.dstack((y, new_u, new_v))
             return new_img
```

Изменив каналы, мы переводим картинку обратно в формат RGB и высчитываем MSE

```
In [80]: img2 = decim(img_yuv, 1)
```

```
In [81]: decoded = yuv2rgb(img2)
         decoded = np.clip(decoded, 0, 1)
         imshow(decoded)
```

```
Out[81]: <matplotlib.image.AxesImage at 0x91718fdb70>
```


```
In [82]: mse(image, decoded)
```

```
Out[82]: 0.038583401505217836
```

Изображения, полученные в ходе работы:

После децимации

Заключение:

В процессе работы были получены базовые навыки обработки цифровых изображений. Изучены различные характеристики изображений такие, как каналы, их яркость, число уровней яркости.

Был написан алгоритм децимации изображения, которая позволяет уменьшить частоты дискретизации дискретного во времени сигнала путём прореживания его отсчётов

Энтропия – информация о том, сколько бит информации нужно для хранения одного пикселя в данном изображении.

Среднеквадратическая ошибка - мера различия двух изображений является среднеквадратическое значение разностного сигнала двух изображений. Иначе, насколько отличается одна картинка от другой.

Таким образом, мы научились работе с изображениями.