Автоконтраст черно-белого изображения

Прочитайте изображение из файла img.png. Примените к нему линейное выравнивание яркости: примените к каждому пикселю функцию

$$f(x) = (x - x_{min}) \cdot \frac{255}{x_{max} - x_{min}}$$

После вычисления функции значения изображения окажутся вещественными. Чтобы привести их к целым числам, используйте метод `img.astype('uint8')`, который возвращает изображение в целых числах. Результат сохраните в файл out_img.png.

В примере входа и выхода указаны ссылки на файлы. Скачав эти файлы, можно протестировать свою программу. Для сравнения вашего ответа с верным используйте функцию `numpy.array_equal`.

Sample Input:

```
https://stepik.org/media/attachments/lesson/58402/tiger-low-contrast.png
```

Sample Output:

```
https://stepik.org/media/attachments/lesson/58402/tiger-high-contrast.png
```

Напишите программу. Тестируется через stdin → stdout

Так точно!

Теперь вам доступен Форум решений , где вы можете сравнить свое решение с другими или спросить совета.

Вы решили сложную задачу, поздравляем! Вы можете помочь другим учащимся в комментариях.

```
from skimage.io import imread, imshow, imsave
img = imread('img.png')
xmax = max(img.ravel())
xmin = min(img.ravel())
k = 255/(xmax-xmin)
ans = ((img-xmin)*k).astype('uint8')
imsave('out_img.png', ans)
```

Следующий шаг | Решить снова

Подсчет минимума и максимума устойчивого автоконтраста

Прочитайте изображение из файла img.png. Подсчитайте минимум и максимум яркости для стабильного автоконтраста этого изображения. Необходимо отбросить 5% самых светлых и 5% самых темных пикселей. Для получения числа отбрасываемых пикселей используйте формулу

$$k = round\left(\#pix \cdot 0.05\right)$$

Два посчитанных числа (минимум и максимум) выведите на стандартный вывод через пробел.

Попробуйте подсчитать минимум и максимум для стабильного автоконтраста двумя способами, указанными в видео.

В примере входа указана ссылка на файлы. Скачав этот файл, можно протестировать свою программу.

Sample Input:

```
https://stepik.org/media/attachments/lesson/58402/tiger-low-contrast.png
```

Sample Output:

```
129 208
```

Здорово, всё верно.

Теперь вам доступен Форум решений, где вы можете сравнить свое решение с другими или спросить совета.

Вы решили сложную задачу, поздравляем! Вы можете помочь другим учащимся в комментариях.

```
from skimage.io import imread, imshow, imsave
img= imread('img.png')
pix = img.shape[0]*img.shape[1]
k= round(pix * 0.05)
v = img.ravel()
v.sort()
xmin = v[k]
xmax = v[pix-k]
print(xmin, xmax)
```

Следующий шаг | Решить снова

Устойчивый автоконтраст черно-белого изображения

Прочитайте изображение из файла img.png. Примените к нему линейное выравнивание яркости: примените к каждому пикселю функцию

$$f(x) = (x - x_{min}) \cdot \frac{255}{x_{max} - x_{min}}$$

Для вычисления максимума и минимума отбрасывайте по 5% самых светлых и самых темных пикселей (как в предыдущем задании). Перед вычислениями приведите изображение в вещественные числа (`img.astype('float')`), иначе может возникнуть переполнение (т.к. значения некоторых пикселей мы игнорируем при подсчете минимума и максимума). После растяжения яркости обрежьте значения изображения от 0 до 255 с помощью функции `numpy.clip` .

После вычисления функции значения изображения окажутся вещественными. Чтобы привести их к целым числам, используйте метод `img.astype('uint8')` , который возвращает изображение в целых числах. Результат сохраните в файл out_img.png.

В примере входа и выхода указаны ссылки на файлы. Скачав эти файлы, можно протестировать свою программу. Для сравнения вашего ответа с верным используйте функцию `numpy.array_equal` .

Sample Input:

`https://stepik.org/media/attachments/lesson/58402/tiger-low-contrast.png`

Sample Output:

`https://stepik.org/media/attachments/lesson/58402/tiger-stable-contrast.png`

Прекрасный ответ.

Теперь вам доступен Форум решений , где вы можете сравнить свое решение с другими или спросить совета.

```
from skimage.io import imread, imshow, imsave
from numpy import *
img = imread('img.png')
imgf=img.astype('float')
pix = imgf.shape[0]*imgf.shape[1]
k= round(pix * 0.05)
v = imgf.ravel()
v.sort()
xmin = v[k]
xmax = v[pix-k]
koef = 255/(xmax-xmin)
f = ((img-xmin)*koef)
ans=clip(f,0,255)
ans1 = ans.astype('uint8')
imsave('out_img.png', ans1)
```

Следующий шаг | Решить снова

Устойчивый цветной автоконтраст

Прочитайте цветное изображение из файла img.png. Примените к нему устойчивый автоконтраст. Для этого:

1. Переведите изображение в вещественные числа от 0 до 1.
2. Переведите изображение в пространство YUV по формулам:

$$\begin{aligned} Y &= \ \ 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B \\ U &= -0.0999 \cdot R - 0.3360 \cdot G + 0.4360 \cdot B \\ V &= \ \ 0.6150 \cdot R - 0.5586 \cdot G - 0.0563 \cdot B \end{aligned}$$

3. Найдите максимум и минимум для устойчивого автоконтраста с отбрасыванием 5% самых светлых и 5% самых темных пикселей.
4. Примените линейное растяжение канала Y по формуле

$$f(x) = (x - x_{min}) \cdot \frac{255}{x_{max} - x_{min}}$$

5. Обрежьте значения канала Y от 0 до 1.
6. Переведите изображение в пространство RGB по формулам:

$$\begin{aligned} R &= Y + 1.2803 \cdot V \\ G &= Y - 0.2148 \cdot U - 0.3805 \cdot V \\ B &= Y + 2.1279 \cdot U \end{aligned}$$

7. Обрежьте значения изображения от 0 до 1.
8. Переведите изображение в целые числа от 0 до 255.

Результат сохраните в файл out_img.png.

В примере входа и выхода указаны ссылки на файлы. Скачав эти файлы, можно протестировать свою программу. Для сравнения вашего ответа с верным используйте функцию `numpy.array_equal`.

Отлично!

Теперь вам доступен Форум решений , где вы можете сравнить свое решение с другими или спросить совета.

Вы решили сложную задачу, поздравляем! Вы можете помочь другим учащимся в комментариях.

```python
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
import numpy as np

img = img_as_float(imread('img.png'))
R, G, B = (img[..., i] for i in range(3))
Y =  0.2126*R+0.7152*G+0.0722*B
U = -0.0999*R-0.3360*G+0.4360*B
V =  0.6150*R-0.5586*G-0.0563*B

vmin, vmax = np.percentile(Y, [5, 95])
Y = np.clip((Y - vmin) / (vmax - vmin), 0, 1)

R = np.clip(Y+1.2803*V, 0, 1)
G = np.clip(Y-0.2148*U-0.3805*V, 0, 1)
B = np.clip(Y+2.1279*U, 0, 1)

img = img_as_ubyte(np.dstack((R, G, B)))
imsave('out_img.png', img)
```

Преобразование серого мира

Прочитайте изображение из файла img.png. Примените к нему преобразование серого мира. Для этого:
1. Сконвертируйте изображение в вещественные числа.
2. Подсчитайте коэффициенты r_w, g_w, b_w как описано в видео.
3. Поделите каналы изображения на коэффициенты.
4. Обрежьте значения пикселей, чтобы они не выходили из допустимого диапазона ([0; 255] или [0;1]).
Результат сохраните в файл out_img.png.

В примере входа и выхода указаны ссылки на файлы. Скачав эти файлы, можно протестировать свою программу. Для сравнения вашего ответа с верным используйте функцию `numpy.array_equal`.

Sample Input:

`https://stepik.org/media/attachments/lesson/60610/railroad.png`

Sample Output:

`https://stepik.org/media/attachments/lesson/60610/railroad-gray-world.png`

Прекрасный ответ.

Теперь вам доступен Форум решений , где вы можете сравнить свое решение с другими или спросить совета.

```
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
import numpy as np

img = img_as_float(imread('img.png'))
R, G, B = (img[..., i] for i in range(3))

avgR=np.mean(R)
avgG=np.mean(G)
avgB=np.mean(B)

avg=(avgR+avgG+avgB)/3

rw=avgR/avg
rg=avgG/avg
rb=avgB/avg

r=np.clip(R/rw,0,1)
g=np.clip(G/rg,0,1)
b=np.clip(B/rb,0,1)

img = img_as_ubyte(np.dstack((r, g, b)))
imsave('out_img.png', img)

```

Выравнивание гистограммы

Слайды видео.

Прочитайте изображение из файла img.png. Примените к нему выравнивание гистограммы по алгоритму, описанному в слайдах и видео. Работать достаточно в целых числах, помещающихся в байт (т.е. изображение конвертировать не нужно). Результат сохраните в файл out_img.png.

В примере входа и выхода указаны ссылки на файлы. Скачав эти файлы, можно протестировать свою программу. Для сравнения вашего ответа с верным используйте функцию `numpy.array_equal`.

Sample Input:

```
https://stepik.org/media/attachments/lesson/60611/landscape.png
```

Sample Output:

```
https://stepik.org/media/attachments/lesson/60611/landscape-histeq.png
```

Так точно!

Теперь вам доступен Форум решений , где вы можете сравнить свое решение с другими или спросить совета.

Вы решили сложную задачу, поздравляем! Вы можете помочь другим учащимся в комментариях.

```
from skimage.io import imread, imsave
from skimage import img_as_float, img_as_ubyte
from numpy import *
from numpy import histogram
from numpy import round
img = imread('img.png')
hist = histogram(img, bins=range(257))[0]
cdf = cumsum(hist)
pix= img.size
minv=cdf[cdf > 0][0]
k=255/(pix-1)
f=round((cdf-minv)*k).astype(uint8)
imsave('out_img.png', f[img])
```