

Дата: 11.06.2023

ФИО: Козлов Евгений Юрьевич

Группа: 224-322

ПРАКТИЧЕСКАЯ РАБОТА

№ 3

Применение гистограммных методов коррекции

Цель: познакомиться с пространственными методами коррекции на примере гистограммной коррекции.

Этапы выполнения

1. Подобрать 2 изображения для коррекции (можно взять из работы 2)
2. Перевести изображения в черно-белое
3. Получить гистограммы изображений
4. Провести нормализацию гистограмм
5. Провести эквализацию гистограмм
6. Провести преобразование гистограммы по произвольно заданной функции распределения

Содержание отчета

1. Название цель работы
2. Используемый язык программирования
3. Параметры исходных изображений (назвать изображения 01 и 02)
 - а. глубина цвета - k, bpp
 - б. размер - m x n, pix
4. Гистограммы изображений

Используемая среда программирования: Visual Studio Code

Используемый язык программирования: Python 3.11.1 64-bit

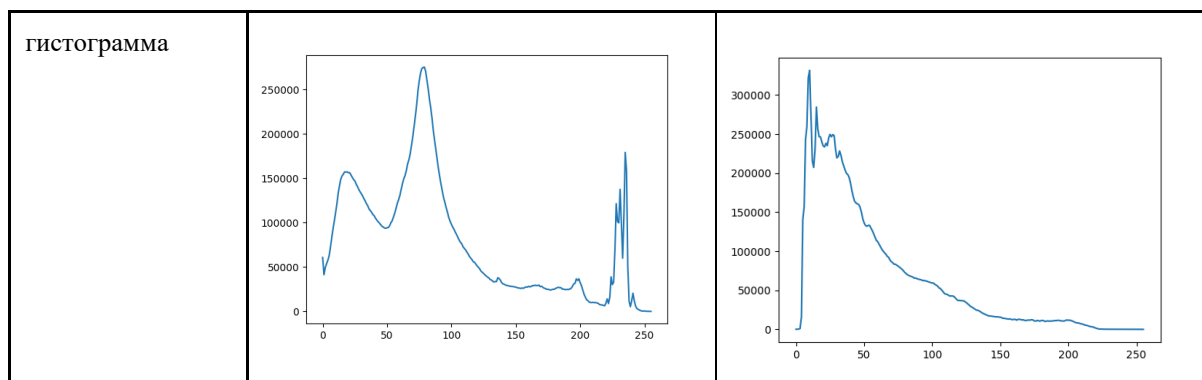
Библиотеки: cv2, matplotlib, numpy, random

Таблица 1

№ изображения	01	02
k,bpp	8	8
m x n, pix	5472 x 3648	5000 x 3333

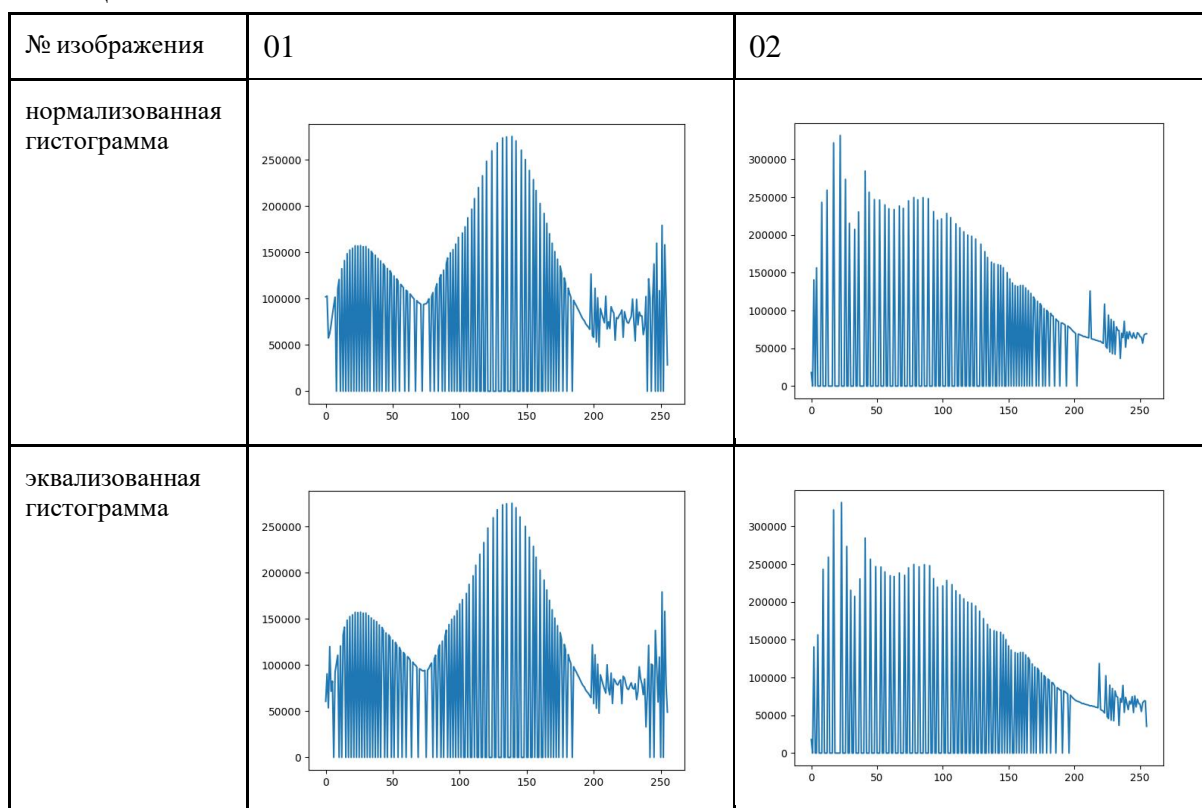
изображение

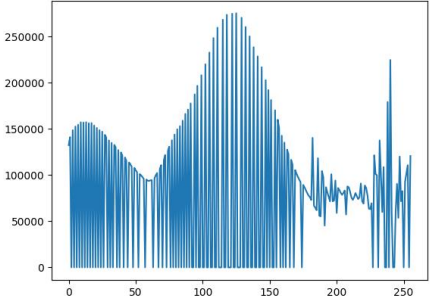
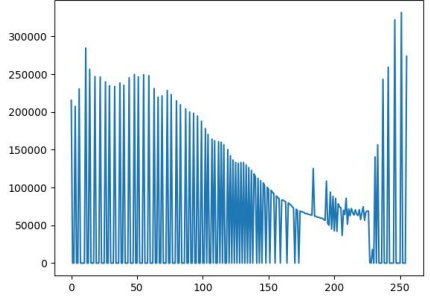




5. Изображения после нормализации
6. Гистограммы нормализованных изображений
7. Изображения после эквализации
8. Гистограммы эквализированных изображений
9. Функция распределения, по которой будет проведено преобразование гистограммы
10. Изображения после применения к гистограмме заданной функции из п.9
11. Гистограммы изображений после преобразования, по заданной функции в п.9

Таблица 2



функция распределения	<pre> cdf_random_1 = hist_before_1.cumsum() cdf_random_1 = (cdf_random_1 - cdf_random_1[rnd_num] + rnd_num)*255/(cdf_random_1[- 1]-1) </pre>	<pre> cdf_random_2 = hist_before_2.cumsum() cdf_random_2 = (cdf_random_2 - cdf_random_2[rnd_num] + rnd_num)*255/(cdf_random_2[- 1]-1) </pre>
гистограмма, преобразованная по заданной функции		

Изображения расположены на диске по ссылке:

https://drive.google.com/drive/folders/1Cbo0_RKFW2fF11GZ7xffq9EVdCxaY0U0?usp=sharing

Код программы:

```

import cv2
import numpy as np
from matplotlib import pyplot as plt
import random

# 1. Подобрать изображения (из практической 2)
INIT_IMG_1 = cv2.imread('img/init/01.jpg')
INIT_IMG_2 = cv2.imread('img/init/02.jpg')

# 2. Перевести изображения в черно-белые
GRAY_IMG_1 = cv2.cvtColor(INIT_IMG_1.copy(), cv2.COLOR_BGR2GRAY)
GRAY_IMG_2 = cv2.cvtColor(INIT_IMG_2.copy(), cv2.COLOR_BGR2GRAY)

cv2.imwrite('img/dist/2/GRAY_IMG_1.png', GRAY_IMG_1)
cv2.imwrite('img/dist/2/GRAY_IMG_2.png', GRAY_IMG_2)

# 3. Вычислить гистограммы
GIST_GRAY_1 = cv2.calcHist([GRAY_IMG_1], [0], None, [256], [0, 256])
GIST_GRAY_2 = cv2.calcHist([GRAY_IMG_2], [0], None, [256], [0, 256])

plt.plot(GIST_GRAY_1)
plt.savefig('img/dist/3/GRAY_IMG_1.png.png')

```

```

plt.close()

plt.plot(GIST_GRAY_2)
plt.savefig('img/dist/3/GRAY_IMG_2.png.png')
plt.close()

# 4. Провести нормализацию гистограмм

# Нормализация гистограммы обеспечивает растяжку не всего диапазона
изменения интенсивностей,
# а только его наиболее информативной части. Под информативной частью
понимается набор пиков
# гистограммы, т.е. интенсивности, которые чаще остальных встречаются
на изображении. Бины,
# соответствующие редко встречающимся интенсивностям, в процессе
нормализации отбрасываются,
# далее выполняется обычная линейная растяжка получившейся гистограммы.

# Вычисление hist
hist_before_1, bins_before_1 = np.histogram(GRAY_IMG_1, 256)
hist_before_2, bins_before_2 = np.histogram(GRAY_IMG_2, 256)

# Вычисление CDF:
# Определение кумулятивной функции распределения (CDF): для непрерывной
функции сумма
# вероятностей появления всех значений, меньших или равных a.  $F(a) = P(x \leq a)$ 
cdf_1 = hist_before_1.cumsum()
cdf_2 = hist_before_2.cumsum()

# Возврат cdf к формату [0,255] (применение формулы распределения)
cdf_1 = (cdf_1-cdf_1[0])*255/(cdf_1[-1]-1)
cdf_2 = (cdf_2-cdf_2[0])*255/(cdf_2[-1]-1)
cdf_1 = cdf_1.astype(np.uint8)
cdf_2 = cdf_2.astype(np.uint8) # Приведение float64 обратно к uint8

# generate img after Histogram Equalization
NORMALIZED_1 = np.zeros((384, 495, 1), dtype=np.uint8)
NORMALIZED_2 = np.zeros((384, 495, 1), dtype=np.uint8)
NORMALIZED_1 = cdf_1[GRAY_IMG_1]
NORMALIZED_2 = cdf_2[GRAY_IMG_2]

GIST_NORM_1, bins_after_1 = np.histogram(NORMALIZED_1, 256)
GIST_NORM_2, bins_after_2 = np.histogram(NORMALIZED_2, 256)

cv2.imwrite('img/dist/4/NORMALIZED_1.png', NORMALIZED_1)
cv2.imwrite('img/dist/4/NORMALIZED_2.png', NORMALIZED_2)

plt.plot(GIST_NORM_1)
plt.savefig('img/dist/4/GIST_NORM_1.png.png')
plt.close()

plt.plot(GIST_NORM_2)

```

```

plt.savefig('img/dist/4/GIST_NORM_2.png.png')
plt.close()

# 5. Провести эквализацию гистограмм

# Целью выравнивания гистограммы (линеаризации, эквализации) является
такое преобразование,
# чтобы, в идеале, все уровни яркости приобрели бы одинаковую частоту,
а
# гистограмма яркостей отвечала бы равномерному закону распределения.

# В OpenCV для этого есть функция cv.equalizeHist (), которая
инкапсулирует процесс
# вычисления переназначения cdf и cdf и создания изображения с
выравниванием гистограммы
# на основе таблицы cdf. Его вход - только изображение в оттенках
серого,
# а выход - это изображение с выравниванием гистограммы.

EQUALIZED_1 = cv2.equalizeHist(GRAY_IMG_1.copy())
EQUALIZED_2 = cv2.equalizeHist(GRAY_IMG_2.copy())

cv2.imwrite('img/dist/5/EQUALIZED_1.png', EQUALIZED_1)
cv2.imwrite('img/dist/5/EQUALIZED_2.png', EQUALIZED_2)

GIST_EQ_1 = cv2.calcHist([EQUALIZED_1], [0], None, [256], [0, 256])
GIST_EQ_2 = cv2.calcHist([EQUALIZED_2], [0], None, [256], [0, 256])

plt.plot(GIST_EQ_1)
plt.savefig('img/dist/5/GIST_EQ_1.png')
plt.close()

plt.plot(GIST_EQ_2)
plt.savefig('img/dist/5/GIST_EQ_2.png')
plt.close()

# 6. Провести преобразование гистограммы по произвольно заданной
функции распределения

cdf_random_1 = hist_before_1.cumsum()
cdf_random_2 = hist_before_2.cumsum()

# cdf_random_2 = (cdf_random_2-cdf_random_2[0])*255/(cdf_random_2[-1]-
1)

# Рандомная ф-ция распределения
rnd_num = random.randint(0, 128)
cdf_random_1 = (cdf_random_1 - cdf_random_1[rnd_num] +
rnd_num)*255/(cdf_random_1[-1]-1)
cdf_random_2 = (cdf_random_2 - cdf_random_2[rnd_num] +
rnd_num)*255/(cdf_random_2[-1]-1)

```

```
cdf_random_1 = cdf_random_1.astype(np.uint8)
cdf_random_2 = cdf_random_2.astype(np.uint8)

NORMALIZED_RND_1 = np.zeros((384, 495, 1), dtype =np.uint8)
NORMALIZED_RND_2 = np.zeros((384, 495, 1), dtype =np.uint8)
NORMALIZED_RND_1 = cdf_random_1[GRAY_IMG_1]
NORMALIZED_RND_2 = cdf_random_2[GRAY_IMG_2]

cv2.imwrite('img/dist/6/NORMALIZED_RND_1.png', NORMALIZED_RND_1)
cv2.imwrite('img/dist/6/NORMALIZED_RND_2.png', NORMALIZED_RND_2)

GIST_NORM_RND_1, bins_rnd_1 = np.histogram(NORMALIZED_RND_1, 256)
GIST_NORM_RND_2, bins_rnd_2 = np.histogram(NORMALIZED_RND_2, 256)

plt.plot(GIST_NORM_RND_1)
plt.savefig('img/dist/6/GIST_NORM_RND_1.png')
plt.close()

plt.plot(GIST_NORM_RND_2)
plt.savefig('img/dist/6/GIST_NORM_RND_2.png')
plt.close()
```