# Дата: 11.06.2023

**ФИО: Козлов Евгений Юрьевич Группа: 224-322**

# ПРАКТИЧЕСКАЯ РАБОТА № 3

Применение гистограммных методов коррекции

**Цель**: познакомится 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)
   1. глубина цвета - k, bpp
   2. 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 |  |  |

|  |  |  |
| --- | --- | --- |
| гистограмма |  |  |

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

Таблица 2

|  |  |  |
| --- | --- | --- |
| № изображения | 01 | 02 |
| нормализованная гистограмма |  |  |
| эквализованная гистограмма |  |  |

|  |  |  |
| --- | --- | --- |
| функция  распределения | 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) | 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) |
| гистограмма, преобразованная по заданной  функции |  |  |

Изображения расположены на диске по ссылке:

<https://drive.google.com/drive/folders/1Cbo0_RKFW2fFl1GZ7xffq9EVdCxaY0U0?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 (а) = Р (х <= а)

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 unit8

# 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()