Министерство науки и высшего образования Российской Федерации НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

# Фундаментальная информатика и информационные технологии

**ОТЧЕТ**

по дисциплине «Распознавание образов»

на тему «Современное состояние систем распознавания образов и компьютерного зрения»

Выполнил

студент группы № 932101

Фам Ань Ту

Проверил

ст. преподаватель

О. Е. Бакланова

Томск–2024

Цель работы: Освоение основных яркостных и геометрических характеристик изображений и их использование для анализа изображений.

Теоретическое обоснование применяемых методов и функций геометрических преобразований:

1) *Гистограммы*.

Гистограмма — это распределение частоты встречаемости пикселей одинаковой яркости на изображении.

Представляет собой одномерный целочисленный массив Hist из 256 элементов [0 . . . 255]. Элементом гистограммы Hist[𝑖] является сумма пикселей изображения с яркостью 𝑖.

Анализируя гистограмму, принимается решение об изменении яркости и(или) контрастности изображения. Позволяет оценить площадь, занимаемую светлыми и темными элементами.

Для RGB-изображения для каждого канала (R, G, B) строится своя гистограмма. Таким образом, можно настроить каждый канал отдельно для результирующего изображения.

Гистограмму можно выровнять, если она неравномерна. Например, с помощью арифметической операции со сдвигом в право и влево соответственно:

Inew(i,j) = I(i,j) + 50 / 255;

Inew(i,j) = I(i,j) - 50 / 255;

Также гистограммы можно растянуть, если интенсивности пикселей областей интереса находятся в узком динамическом диапазоне. Применяется выражение:

I\_new = ((I – I\_min) / (I\_max – I\_min)) ^ alpha,

где I и I\_new –массивы значений интенсивности исходного и нового изображений,

𝐼𝑚𝑖𝑛 и 𝐼𝑚𝑎𝑥 — минимальное и максимальное значения интенсивностей исходного изображения соответственно; alpha — коэффициент нелинейности.

Преобразование нелинейно, так как если бы alpha = 1, желаемого эффекта не будет достигнуто.

2) *Проекции*.

Проекцией на ось называется сумма интенсивностей пикселей изображений, взятая в направлении, перпендикулярном данной оси.

Простейшим случаем проекции двумерного изображения являются вертикальная проекция на ось 𝑂𝑥, представляющая собой сумму интенсивностей пикселей по столбцам изображения:

Proj 𝑋(𝑦) = dim ∑ 𝑌 −1 𝑦=0 𝐼(𝑥,𝑦),

и горизонтальная проекция на ось 𝑂𝑦, представляющая собой сумму интенсивностей пикселей по строкам изображения:

Proj 𝑌 (𝑥) = dim ∑ 𝑋−1 𝑥=0 𝐼(𝑥,𝑦).

Проекция изображения на ось 𝑂𝑒 определяется следующим выражением:

Proj 𝐸(𝑡) = ∑𝑥𝑒\_𝑥+𝑦𝑒\_𝑦=𝑡 𝐼(𝑥,𝑦).

Анализ массива проекции позволяет выделять характерные точки функции проекции, которые соответствуют контурам объектов на изображении. Например, если на изображении имеются контрастные объекты, то в проекции будут видны перепады или экстремумы функции, соответствующие положению каждого из объектов.

3) *Профили*.

Профиль вдоль линии — это функция интенсивности изображения, распределенного вдоль данной линии (прорезки).

Профиль строки изображения:

Profile 𝑖(𝑥) = 𝐼(𝑥,𝑖), где 𝑖 — номер строки изображения 𝐼.

Профиль столбца изображения:

Profile 𝑗(𝑦) = 𝐼(𝑗,𝑦), где 𝑗 — номер столбца изображения 𝐼.

Профиль можно рассматривать вдоль любой прямой, ломаной или кривой линии, пересекающей изображение. После формирования массива профиля изображения проводится его анализ.

Можно выделить особые точки функции профиля, соответствующие контурам изображения, пересекаемым данной линией.

1.1 Гистограммы.

Исходное изображение:



Листинг построения гистограмм:

import numpy as np

import matplotlib.pyplot as plt

import cv2, sys

image = cv2.imread('Pic6.jpg')

gray\_img = (0.1140\*image[:,:,0] + 0.5870\*image[:,:,1]+ 0.2989\*image[:,:,2]).astype(np.uint8)

w, h = gray\_img.shape

plt.axis("off")

plt.imshow(gray\_img, 'gray')

plt.title('Original Image')

hist\_f = np.zeros(256) # Creat 1D array of zeros

for i in range(w):

    for j in range(h):

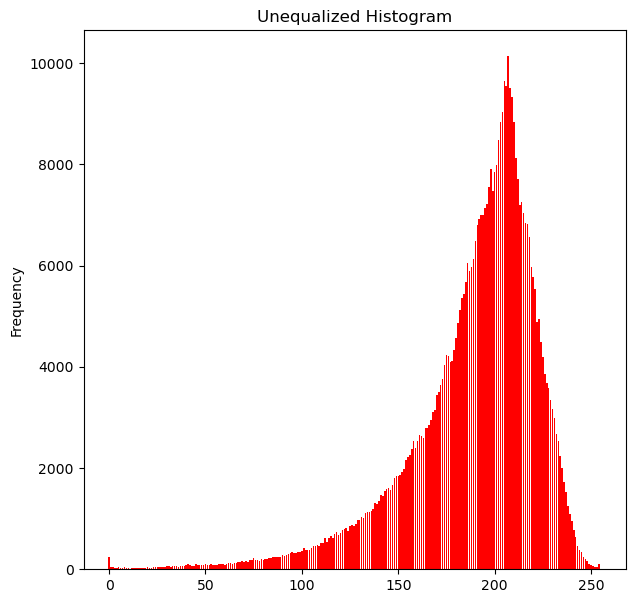
        hist\_f[(gray\_img[i,j])] = hist\_f[(gray\_img[i,j])] + 1

plt.figure(figsize=(7, 7))

plt.bar(list(range(len(hist\_f))), hist\_f, color='red')

plt.title('Unequalized Histogram')

plt.ylabel('Frequency')



1.2 Выравнивание гистограммы.

r\_k = np.unique(gray\_img)

p\_k = hist\_f

prob\_m = p\_k / np.sum(p\_k)

C\_m = np.zeros(len(p\_k))

for i in range(len(p\_k)):

    C\_m[i] = prob\_m[i] + C\_m[i-1]

L = C\_m \* np.max(gray\_img)

L\_r = np.round(L, 0)

eq\_img = np.zeros((w, h), np.uint8)

for i in range(w):

    for j in range(h):

        '''

        This is just like the above done hist\_f loop

        '''

        eq\_img[i, j] = L\_r[gray\_img[i, j]]

plt.figure(figsize=(7, 7))

plt.imshow(eq\_img, 'gray')

plt.axis("off")

plt.title('Equalized Image')



eq\_hist = np.zeros(256)

for i in range(w):

    for j in range(h):

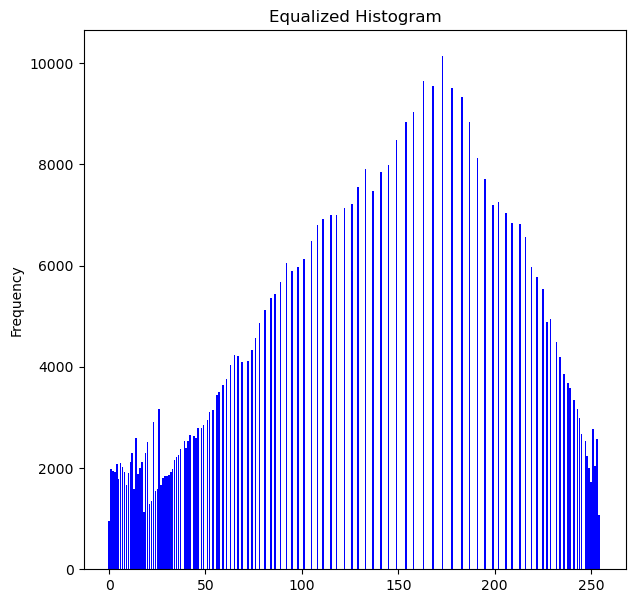
        eq\_hist[(eq\_img[i,j])] = eq\_hist[(eq\_img[i,j])] + 1

plt.figure(figsize=(7, 7))

plt.bar(list(range(len(eq\_hist))), eq\_hist, color='blue')

plt.title('Equalized Histogram')

plt.ylabel('Frequency')



1.2 Растяжение гистограммы

import cv2

import numpy as np

import matplotlib.pyplot as plt

img =  cv2.imread('Pic6.jpg')

# The values of xp and fp can be varied to create custom tables as required

# and it will stretch the contrast even if min and max pixels are 0 and 255

xp = [0, 64, 128, 192, 255]

fp = [0, 16, 128, 240, 255]

x = np.arange(256)

table = np.interp(x, xp, fp).astype('uint8')

# cv2.LUT will replace the values of the original image with the values in the

# table. For example, all the pixels having values 1 will be replaced by 0 and

# all pixels having values 4 will be replaced by 1.

img = cv2.LUT(img, table)

plt.imshow(img)

plt.axis("off")

plt.title('Contrast Stretched Image')

# from shujaat

def contrast\_stretching(z, a, b, z1, zk):

    new\_array = np.copy(z)

    for i,value in enumerate(z):

        if value>=a and value<=b:

            new\_pixel\_value = (((zk - z1)/(b-a))\*value) + ((z1\*b - zk\*a)/(b-a))

            new\_array[i] = new\_pixel\_value

    return new\_array



2.1 Проекции

Исходное изображение:



import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = 'Pic1.jpg'

I = cv2.imread(image\_path)

I\_ = cv2.cvtColor(I, cv2.COLOR\_BGR2RGB)

plt.imshow(I\_)

plt.axis('off')

plt.show()

if I is None:

    print("Ошибка при загрузке изображения")

else:

    # Преобразуем изображение в градации серого (если изображение цветное)

    if I.ndim == 3:  # Проверка, если изображение имеет 3 канала (BGR)

        gray\_image = cv2.cvtColor(I, cv2.COLOR\_BGR2GRAY)

    else:

        gray\_image = I

    # Вычисление проекций на ось Oy (вертикальная проекция)

    proj\_Oy = np.sum(gray\_image, axis=1) / 255

    # Вычисление проекций на ось Ox (горизонтальная проекция)

    proj\_Ox = np.sum(gray\_image, axis=0) / 255

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    # Вертикальная проекция (Oy)

    ax1.plot(proj\_Oy, range(len(proj\_Oy)), color='black')

    ax1.set\_title('Проекция на Oy (Вертикальная)')

    ax1.set\_xlabel('Интенсивность')

    ax1.set\_ylabel('Oy')

    ax1.invert\_yaxis()  # Инвертируем ось Y для корректного отображения

    # Горизонтальная проекция (Ox)

    ax2.plot(range(len(proj\_Ox)), proj\_Ox, color='black')

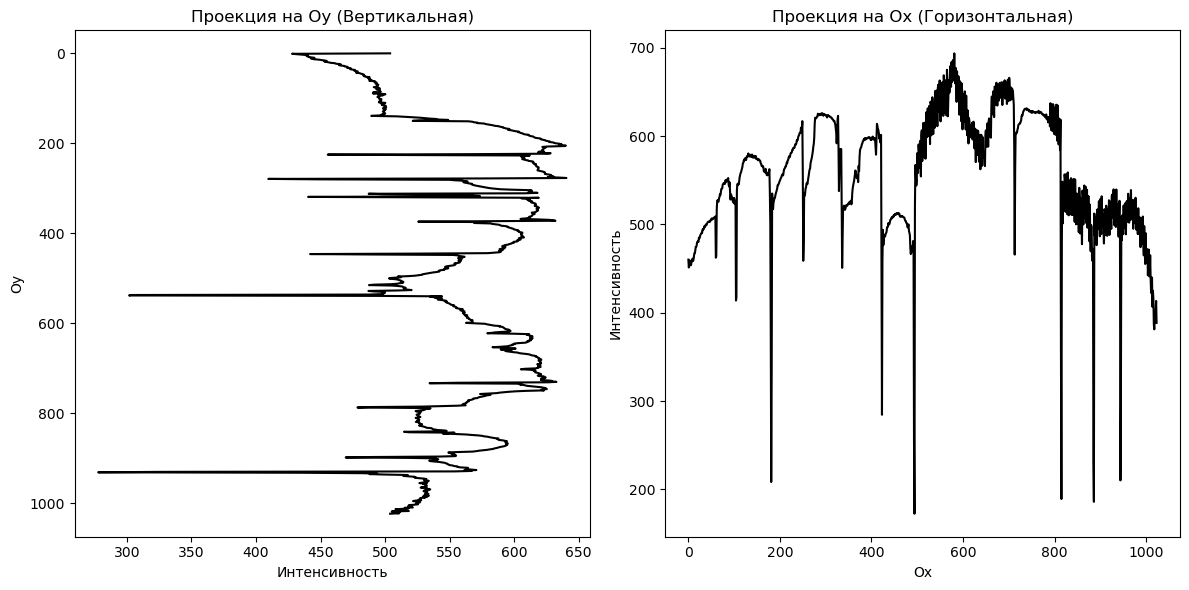
    ax2.set\_title('Проекция на Ox (Горизонтальная)')

    ax2.set\_xlabel('Ox')

    ax2.set\_ylabel('Интенсивность')

    plt.tight\_layout()

    plt.show()



3.1 Профили



import cv2

import matplotlib.pyplot as plt

import numpy as np

# Шаг 1: Загрузка изображения штрих-кода

image\_path = 'Pic3.jpg'  # Замените на путь к вашему изображению

I = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  # Загрузим изображение в градациях серого

# Проверяем успешность загрузки

if I is None:

    print(f"Ошибка: изображение не удалось загрузить. Проверьте путь: {image\_path}")

else:

    # Шаг 2: Определим строку изображения для построения профиля (например, центральная строка)

    row\_number = I.shape[0] // 2  # Центральная строка

    profile = I[row\_number, :]  # Профиль строки

    plt.imshow(cv2.cvtColor(I, cv2.COLOR\_BGR2RGB))

    plt.title('Исходное изображение штрих-кода')

    plt.axis('off')

    plt.show()

    # Шаг 3: Построение профиля штрих-кода

    plt.figure(figsize=(10, 4))

    plt.plot(profile, color='black')  # Отобразим профиль

    plt.title('Профиль изображения вдоль штрих-кода')

    plt.xlabel('Позиция пикселя вдоль оси X')

    plt.ylabel('Интенсивность (яркость)')

    plt.show()



4. Выводы.

С помощью библиотеки OpenCV можно обрабатывать изображения, исследовать их характер, различные параметры (яркость, контрастность), используя гистограммы, профили и проекции.

Эту библиотеку поддерживают Python, C++, MatLab.

Для работы с гистограммами к первому входному изображению было применен сдвиг гистограммы каналов:

- сдвиг синего вправо, так как множество значений находилось слева,

- сдвиг красного влево, так как множество значений находилось справа.

Также было применено нелинейное растяжение гистограммы, что позволило получить более сглаженное изображение.

При построении проекции для второго входного изображения был изучен характер расположения объектов, проанализированы пустые области.

Для изображения штрих-кода был построен профиль вдоль штрих-кода для анализа его структуры.