|  |  |
| --- | --- |
| тов_знак_прав | МИНОБРНАУКИ РОССИИ  федеральное государственное автономное образовательное учреждение  высшего образования  «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»  **(СПбГЭТУ «ЛЭТИ»)** |

Кафедра САУ

ОТЧЕТ

по лабораторно-практической работе № 4

По курсу: «Техническое зрение»

Тема: Пороговые фильтры.

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6491 |  | Зверев Г. Ю.  Михайленко Д. М. |
| Преподаватель |  | Моклева К. А. |

Санкт-Петербург

2020 г.

Цель работы: изучить принцип применения пороговых фильтров для

обработки изображений.

Основные задания:

1) Напишите свою “легкую” реализацию cv2.threshold() только для

варианта THRESH\_BINARY. Функция должна принимать значение threshold. Пусть maxVal по умолчанию всегда будет 255.

2) Примените на практике бинаризацию всех типов для следующих

изображений (нескольких каждого типа):

a) фото дорожной разметки белого цвета;

b) изображение отсканированного текстового документа;

c) фото написанного от руки или напечатанного текста.

Для каждого типа опишите, какой результат был получен.

Дополнительное задание:

В данном задании вам будет выдан набор фотографий, у которых все пиксели имеют яркость не ниже, чем некоторое значение A, и не выше, чем некоторое другое значение B. Вашей задачей будет преобразовать цвета на изображении так, чтобы гистограмма этого изображения растянулась на весь диапазон яркости [0, 255]. Заметьте, что для этого достаточно определить по изображению значения A и B, а затем диапазон яркости [A, B] пропорционально растянуть до диапазона [0, 255]. Формулу для растягивания диапазона вам предлагается вывести самостоятельно.

Основное задание №1

*Исходный код:*

from cv2 import cv2

# реализация своей функции "cv2.treshold()"

picture = 'Putin.png'

maxValue = 255

def treshold(image, Value):

img = cv2.imread(image, cv2.IMREAD\_GRAYSCALE)

cv2.imshow('img before', img)

for i in range(len(img)):

for j in range(len(img[1])):

if img[i, j] > 128:

img[i, j] = Value

else:

img[i, j] = 0

cv2.imshow('img after', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

treshold(picture, maxValue)

*Результат:*





Основное задание №2

*Исходный код:*

from cv2 import cv2

'''

Применить на практике бинаризацию всех типов для изображений.

Подобрано по 2 примера изображений на каждый тип, иллюстрирующие разницу результатов использованных методов.

'''

"""Для дорог лучше подходит метод TRESH\_TOZERO, т.к. картинка монотонна, а полученное изображение четчё при ручном подборе трешхолда"""

img = cv2.imread('Doroga1.png', cv2.IMREAD\_GRAYSCALE)

treshold, img1 = cv2.threshold(img, thresh = 200, maxval = 255, type = cv2.THRESH\_TOZERO)

treshold, img2 = cv2.threshold(img, 200, 255, cv2.THRESH\_OTSU)

img3 = cv2.adaptiveThreshold(img, maxValue = 255, adaptiveMethod = cv2.ADAPTIVE\_THRESH\_MEAN\_C, treshholdType = cv2.THRESH\_BINARY, blockSize = 11, C = -4)

img\_2 = cv2.imread('Doroga2.png', cv2.IMREAD\_GRAYSCALE)

treshold, img4 = cv2.threshold(img\_2, 200, 255, cv2.THRESH\_TOZERO)

treshold, img5 = cv2.threshold(img\_2, 200, 255, cv2.THRESH\_OTSU)

img6 = cv2.adaptiveThreshold(img\_2, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 3, -4)

"""Для сканов лучше подходит как адаптивный метод, так и TRESH\_OTSU, т.к. скан документа не монотонный"""

# img = cv2.imread('Scan1.png', cv2.IMREAD\_GRAYSCALE)

# treshold, img1 = cv2.threshold(img, 128, 255, cv2.THRESH\_TOZERO)

# treshold, img2 = cv2.threshold(img, 128, 255, cv2.THRESH\_OTSU)

# img3 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 31, 30)

# img\_2 = cv2.imread('Scan2.png', cv2.IMREAD\_GRAYSCALE)

# treshold, img4 = cv2.threshold(img\_2, 128, 255, cv2.THRESH\_TOZERO)

# treshold, img5 = cv2.threshold(img\_2, 128, 255, cv2.THRESH\_OTSU)

# img6 = cv2.adaptiveThreshold(img\_2, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 29, 30)

"""Для фото написанного от руки также подходит лучше адаптивный, так как происходят резкие изменения уровня яркости на небольших расстояниях"""

# img = cv2.imread('Text1.png', cv2.IMREAD\_GRAYSCALE)

# treshold, img1 = cv2.threshold(img, 128, 255, cv2.THRESH\_TOZERO)

# treshold, img2 = cv2.threshold(img, 128, 255, cv2.THRESH\_OTSU)

# img3 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 31, 15)

# img\_2 = cv2.imread('Text2.png', cv2.IMREAD\_GRAYSCALE)

# treshold, img4 = cv2.threshold(img\_2, 128, 255, cv2.THRESH\_TOZERO)

# treshold, img5 = cv2.threshold(img\_2, 128, 255, cv2.THRESH\_OTSU)

# img6 = cv2.adaptiveThreshold(img\_2, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY, 29, 15)

cv2.imshow('Tresh\_tozero', img1)

cv2.imshow('Tresh\_otsu', img2)

cv2.imshow('Tresh\_adaptive', img3)

cv2.imshow('Tresh\_tozero2', img4)

cv2.imshow('Tresh\_otsu2', img5)

cv2.imshow('Tresh\_adaptive2', img6)

cv2.waitKey(0)

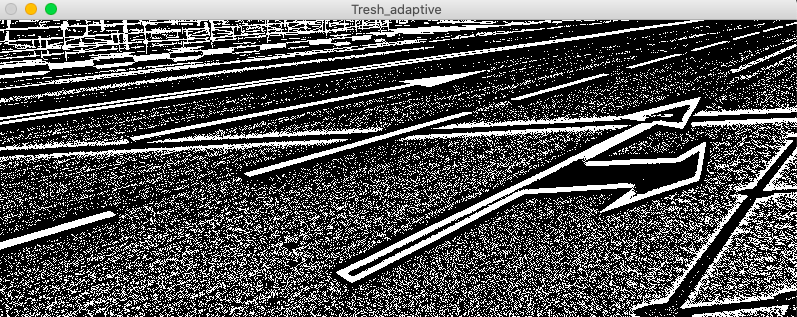
cv2.destroyAllWindows()

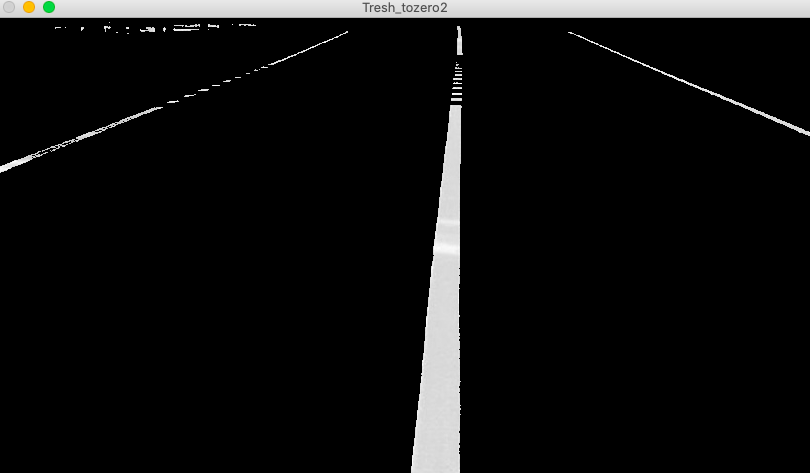
*Результат:*

Изображения дорожной разметки.

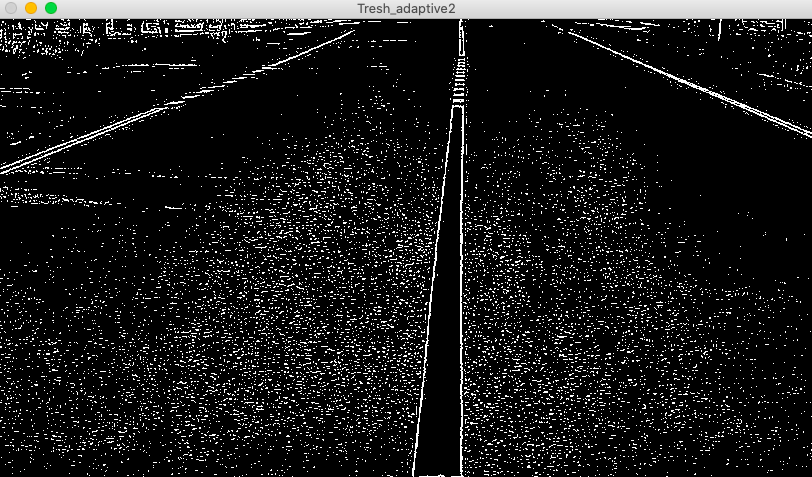






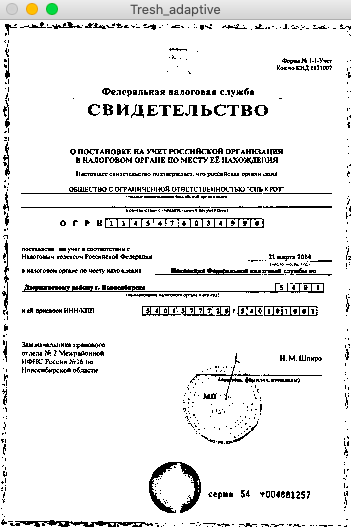


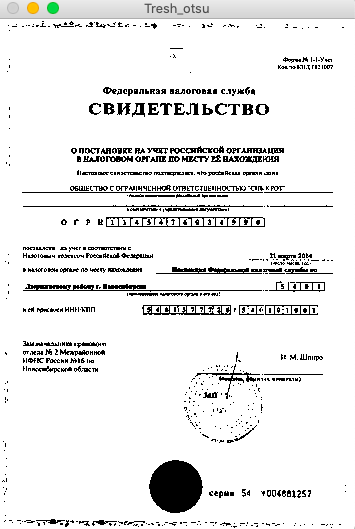


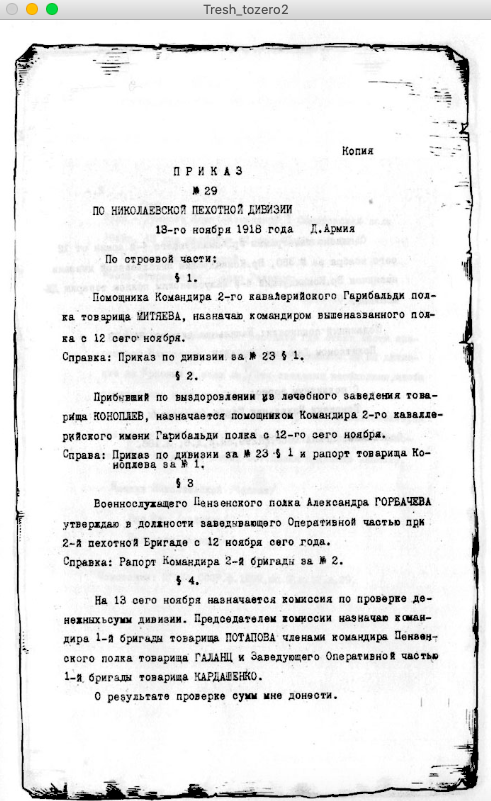


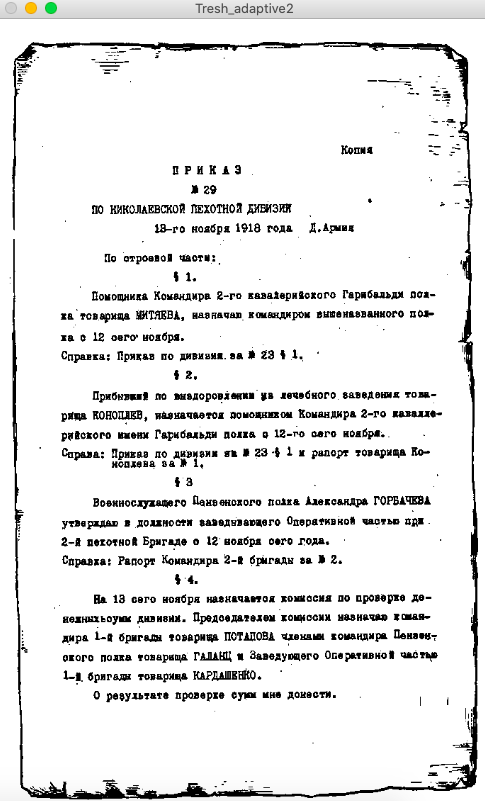
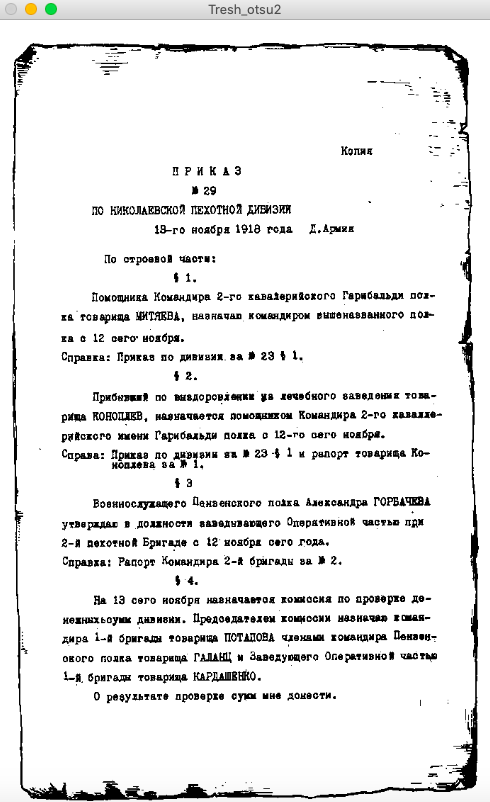
Для изображений дорожной разметки лучше всего подходит метод TRESH\_TOZERO, так как изображения монотонны, нет резких изменений уровня яркости + удобство в ручном подборе порогового значения(threshold).

Изображения сканов документов.



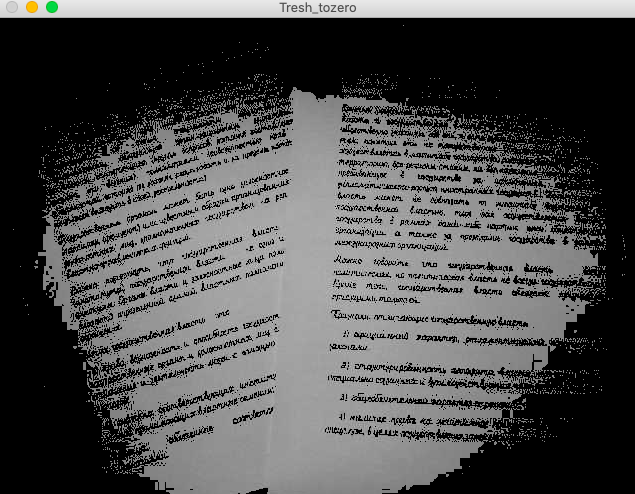


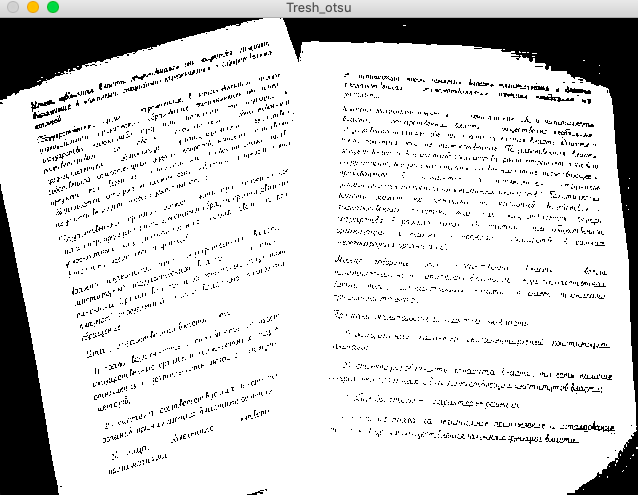


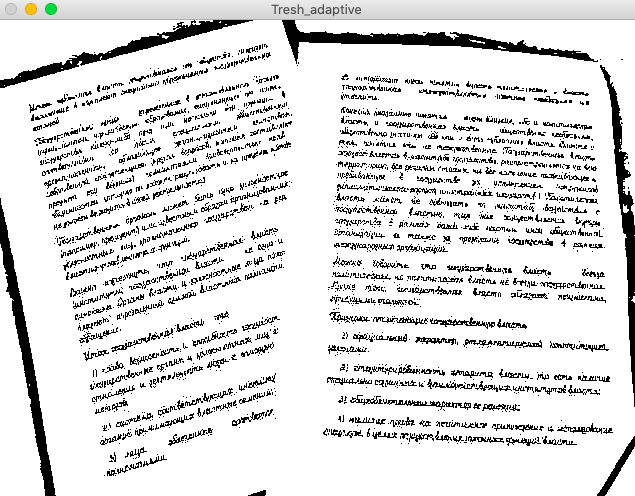


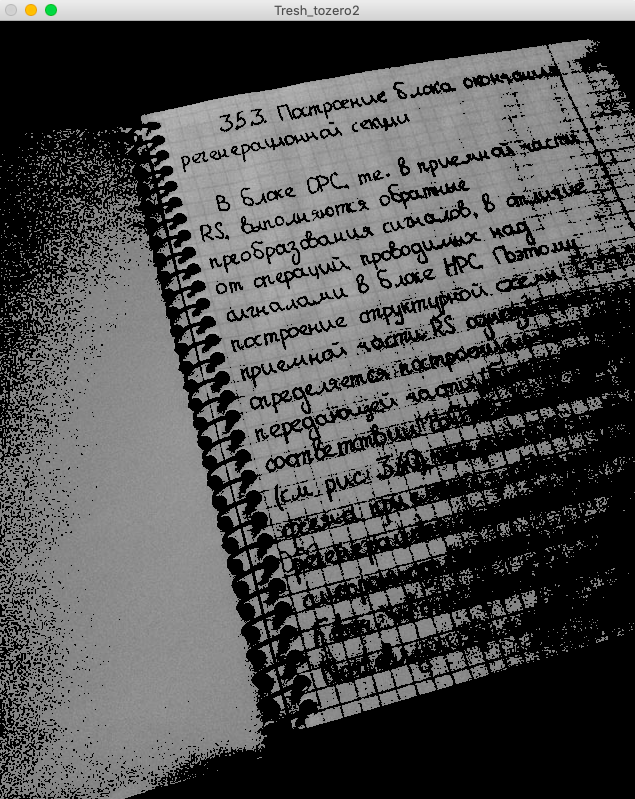
По результатам применения бинаризации можно сказать, что для сканов лучше подходят как адаптивный метод, так и TRESH\_OTSU, т.к. скан документа не монотонный и имеет резкие изменения яркости изображения, с которыми трудно справиться методу TRESH\_TOZERO.

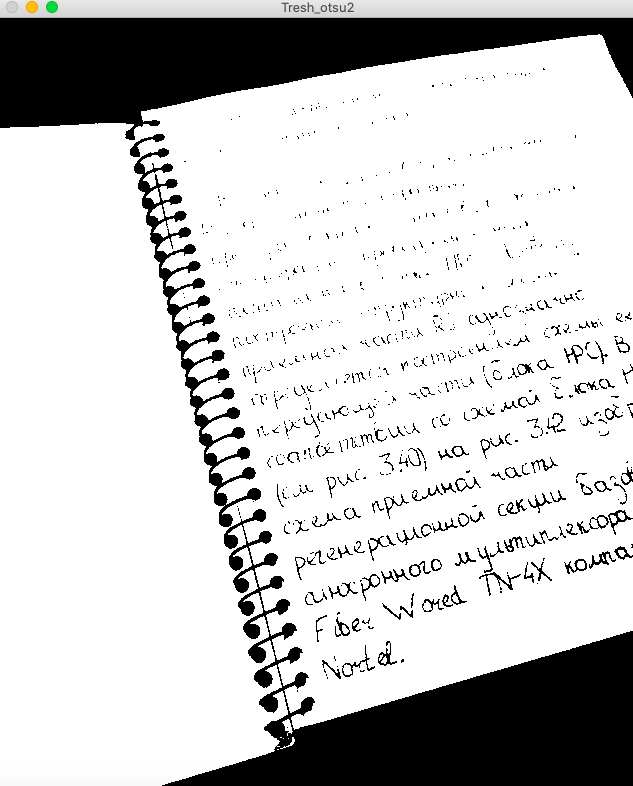
Фото написанного от руки текста.

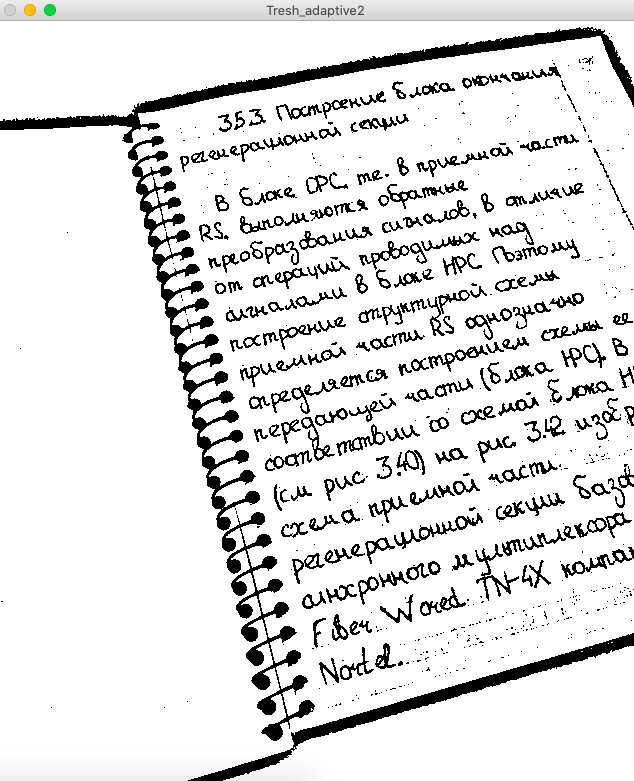












Изображения имеют резкие изменения уровня яркости, лучший метод бинаризации – адаптивный.

Дополнительное задание

*Исходный код:*

from cv2 import cv2

# Изображение 1

lr41 = cv2.imread('LR\_4\_1.png', cv2.IMREAD\_REDUCED\_GRAYSCALE\_2)

print(type(lr41))

print(lr41)

max\_ind = lr41.max().max()

min\_ind = lr41.min().min()

for i in range(len(lr41)):

for j in range(len(lr41[1])):

lr41[i,j] = lr41[i,j] - min\_ind

lr41[i,j] = lr41[i,j] / (max\_ind - min\_ind) \* 255

# Изображение 2

lr42 = cv2.imread('LR\_4\_2.png', cv2.IMREAD\_REDUCED\_GRAYSCALE\_2)

max\_ind = lr42.max().max()

min\_ind = lr42.min().min()

for i in range(len(lr42)):

for j in range(len(lr42[1])):

lr42[i,j] = lr42[i,j] - min\_ind

lr42[i,j] = lr42[i,j] / (max\_ind - min\_ind) \* 255

# Изображение 3

lr43 = cv2.imread('LR\_4\_3.png', cv2.IMREAD\_REDUCED\_GRAYSCALE\_2)

max\_ind = lr43.max().max()

min\_ind = lr43.min().min()

for i in range(len(lr43)):

for j in range(len(lr43[1])):

lr43[i,j] = lr43[i,j] - min\_ind

lr43[i,j] = lr43[i,j] / (max\_ind - min\_ind) \* 255

cv2.imshow('LR\_4\_1\_FIX.png', lr41)

cv2.imshow('LR\_4\_2\_FIX.png', lr42)

cv2.imshow('LR\_4\_3\_FIX.png', lr43)

cv2.waitKey(0)

cv2.destroyAllWindows()

Самым интересным примером является изображение LR\_4\_3.png, так как у него смещена гистограмма относительно центра, приведем результат работы программы только для этого изображения.

До



После



Вывод: в ходе лабораторной работы было изучено применение порогового фильтра к изображению, также научились повышать контрастность изображения. Цвета изображения рекомендуется изменять таким образом, чтобы гистограмма пропорционально растягивалась на весь диапазон яркости. Дополнительно были использованы ранее не изученные методы библиотек OpenCV и NumPy.