# Дата: 27.03.2023

**ФИО: Леонов Владислав Денисович Группа: 224-322**

# ПРАКТИЧЕСКАЯ РАБОТА № 2

**Применение методов градационной коррекции по переходным кривым**

# Цель работы

Познакомится с пространственными методами коррекции на примере градационной коррекции по переходным кривым.

# Содержание работы Этапы выполнения:

1. Подобрать 2 изображения для коррекции
2. Перевести изображения в черно-белое
3. Преобразовать изображения в негатив
4. Провести логарифмическое преобразование
5. Провести степенное преобразование с γ>1, γ<1
6. Провести кусочно-линейное преобразование
7. Провести вырезание уровней в изображении (для одного изображения)

# 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. Изображение, преобразованное в негатив (01\_neg, 02\_neg)
5. Вид функции преобразования
6. Параметры логарифмических преобразований
7. Изображение после логарифмического преобразования (01\_log\_x, 01\_log\_y, 02\_log\_x, 02\_log\_y)
8. Вид функций преобразования
9. Параметры степенных преобразований
10. Изображение после степенного преобразования (01\_deg\_x, 01\_deg\_y, 02\_deg\_x, 02\_deg\_y)
11. Вид функций преобразования
12. Параметры кусочно-линейного преобразования
13. Изображения после кусочно-линейного преобразования (01\_sl, 02\_sl)
14. Вид функций преобразования
15. Номера вырезаемых уровней в выбранном изображении
16. Изображения вырезанных уровней
17. Приложить код программы

Исходные изображения и все изображения после коррекций выложить на облачное хранилище и приложить ссылку.

1. **Исходные данные и программное обеспечение** Используемая среда программирования: Visual Studio Code Используемый язык программирования: Python 3.7.8rc1 64-bit Используемые библиотеки:

cv2 – <https://opencvguide.readthedocs.io/en/latest/opencvpython/basics.html> NumPy - работа с массивами

# Выполнение работы

1. Подобрать 2 изображения для коррекции Подобранные изображения:



1. Перевести изображения в черно-белое

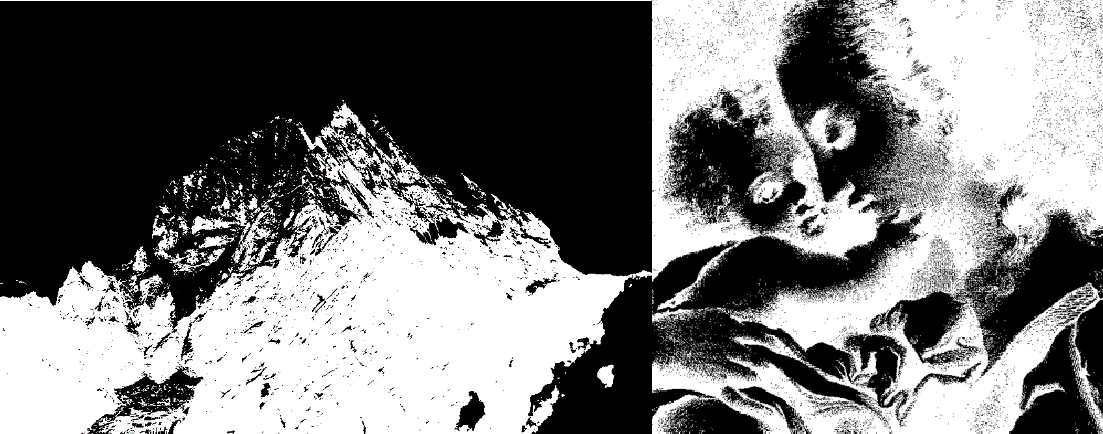
Изображения в черно-белом на основе яркости пространства LAB:



В cv2 параметр L в пространстве LAB имеет значение от 0 до 255, поэтому если параметр L > (255 / 2), то выставляется значение 255, иначе 0.

1. Преобразовать изображения в негатив

Изображения в негативе на основе яркости пространства LAB:



Изображения в негативе на основе инвертированных пикселей пространства RGB:



Для преобразования в негатив, используется функция s=255-r, где s –

результат преобразования, r – исходное изображение, в данной функции происходит инвертирование пикселей на изображении, если они были белыми, то станут черными.

1. Провести логарифмическое преобразование

Изображения после логарифмического преобразования на основе пространства RGB с значением с = 255:



Изображения после логарифмического преобразования на основе

пространства RGB с значением с = 100:



Вид функции преобразования выглядит так:

s = c \* (np.log(1 + r)/(np.log(1 + np.max(r))))

Параметры: r – входное изображение (входные пиксели), c – константа масштабирования.

Здесь np.log - это логарифм по основанию e (натуральный логарифм), а np.max - это функция, которая возвращает максимальное значение в массиве r. Формула берет логарифм от каждого элемента массива r, затем нормализует значения таким образом, чтобы максимальное значение было равно 1. Затем она умножает результат на c, чтобы получить значения в диапазоне от 0 до c.

1. Провести степенное преобразование с γ>1, γ<1

Изображения после степенного преобразования на основе пространства RGB с гаммой γ = 1.5:



Изображения после степенного преобразования на основе пространства RGB с гаммой γ = 0.5:



Вид функции преобразования выглядит так:

s=c\*r^y

В функции преобразования сначала строится таблица соответствия пикселей по формуле s=c\*r^y, где c – константа, r – входной пиксель, y – константа гаммы, s – выходной пиксель.

После с помощью функции LUT от cv2 происходит замена пикселей на те, которые указаны в таблице соответствия.

Параметры: r – входное изображение (входные пиксели), y – константа гаммы, если она больше 1, то пиксели будут затемняться, если меньше, то будут осветляться.

1. Провести кусочно-линейное преобразование

Изображения после кусочно-линейного преобразования на основе пространства RGB с параметрами (55, 0, 210, 255):



Вид функции преобразования выглядит так:

if x < r1:

return (s1 / r1) \* x elif r1 <= x < r2:

return ((s2 - s1) / (r2 - r1)) \* (x - r1) + s1

else:

return ((255 - s2) / (255 - r2)) \* (x - r2) + s2

Параметры: r – входное изображение (пиксели), r1 – параметр первой границы, например, до 55, s1 – результирующий параметр первой границы, например, 0, означает, что результирующие пиксели станут 0, r2 – параметр второй границы, s2 – результирующий параметр второй границы.

1. Провести вырезание уровней в изображении (для одного изображения)

Изображения первой картинки после вырезания 3-х каналов в отдельные файлы на основе RGB:



Номера вырезаемых каналов: 0, 1, 2;

Изображения первой картинки после вырезания 3-х уровней в отдельные файлы на основе RGB:



Номера вырезаемых каналов: 4, 5, 6;

# Вывод:

Проведя пространственные методы коррекции на примере градационной коррекции по переходным кривым было обнаружено, что при преобразовании изображений в ч/б, на первом изображении черным участком становится гора, а на втором изображении становятся черными тени, при инвертировании все пиксели меняются на противоположные, то есть белые становятся черными, а черные белыми. Логарифмические преобразования делают изображение более светлым. Степенное преобразование с гаммой больше 1 делают изображение более темным, а с менее 1 более светлым. Кусочно-линейное преобразование позволяет делать одни участки более темными, а другие более светлыми.

Все изображения хранятся в облаке по адресу: <https://disk.yandex.ru/d/vR-GnmwpwIEEAQ>

# Код программы:

# https://translated.turbopages.org/proxy\_u/en-ru.ru.113ca748-64289592-86f00d49- 74722d776562/[https/www.geeksforgeeks.org/P](http://www.geeksforgeeks.org/PYTHON-INTENSITY-TRANSFORMATION-)YTHON-[INTENSITY-TRANSFORMATION-](http://www.geeksforgeeks.org/PYTHON-INTENSITY-TRANSFORMATION-)

OPERATIONS-ON-IMAGES/

import cv2

import numpy as np

# \_1\_ Прочитайте изображение в формате LAB

\_source\_img\_1 = cv2.imread('\_images/\_1.jpg')

\_source\_img\_2 = cv2.imread('\_images/\_2.jpg')

\_lab\_color\_img\_1 = cv2.cvtColor(\_source\_img\_1.copy(), cv2.COLOR\_BGR2LAB)

\_lab\_color\_img\_2 = cv2.cvtColor(\_source\_img\_2.copy(), cv2.COLOR\_BGR2LAB) cv2.imwrite('\_images/results/\_1\_lab\_color\_img\_1.png', \_lab\_color\_img\_1) cv2.imwrite('\_images/results/\_1\_lab\_color\_img\_2.png', \_lab\_color\_img\_2)

# \_2\_ Преобразуйте изображение в черно-белое def blackwhite\_from\_lab(\_lab\_color\_img):

width, height, depth = \_lab\_color\_img.shape for i in range(width):

for j in range(height):

l, a, b = \_lab\_color\_img[i, j] gray = 255 if (l > 255 / 2) else 0

\_lab\_color\_img[i, j] = (gray, gray, gray) return \_lab\_color\_img

\_blackwhite\_from\_lab\_img\_1 = blackwhite\_from\_lab(\_lab\_color\_img\_1.copy())

\_blackwhite\_from\_lab\_img\_2 = blackwhite\_from\_lab(\_lab\_color\_img\_2.copy()) cv2.imwrite('\_images/results/\_2\_blackwhite\_from\_lab\_img\_1.png',

\_blackwhite\_from\_lab\_img\_1) cv2.imwrite('\_images/results/\_2\_blackwhite\_from\_lab\_img\_2.png',

\_blackwhite\_from\_lab\_img\_2)

# \_3\_ Преобразуйте изображение в негатив

negativ\_from\_blackwhite\_lab\_img\_1 = 255 - \_blackwhite\_from\_lab\_img\_1.copy() negativ\_from\_blackwhite\_lab\_img\_2 = 255 - \_blackwhite\_from\_lab\_img\_2.copy() cv2.imwrite('\_images/results/\_3\_negativ\_from\_blackwhite\_lab\_img\_1.png', negativ\_from\_blackwhite\_lab\_img\_1) cv2.imwrite('\_images/results/\_3\_negativ\_from\_blackwhite\_lab\_img\_2.png', negativ\_from\_blackwhite\_lab\_img\_2)

# Негатив RGB

\_negativ\_from\_rgb\_img\_1 = 255 - \_source\_img\_1.copy()

\_negativ\_from\_rgb\_img\_2 = 255 - \_source\_img\_2.copy() cv2.imwrite('\_images/results/\_3\_negativ\_from\_rgb\_img\_1.png',

\_negativ\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_3\_negativ\_from\_rgb\_img\_2.png',

\_negativ\_from\_rgb\_img\_2)

# \_4\_ Логарифмическое преобразование изображения s = c log(1+r), где c - константа = 1

"""

Логарифмическое преобразование изображения - это метод обработки изображений, который используется для улучшения контраста изображения.

Оно расширяет темные пиксели изображения по сравнению с более высокими значениями пикселей.

Формула для применения логарифмического преобразования к изображению выглядит следующим образом:

S = c \* log (1 + r),

где R - значение пикселя входного изображения,

C - константа масштабирования и S - значение пикселя выходного изображения. """

# RGB

def get\_log\_from\_rgb(r, c):

# Формула лог. преобразования

\_max = np.max(r) # Значение 255

\_log\_max = np.log(1 + \_max) # Значение 5.54 s = c \* (np.log(1 + r)/\_log\_max)

# 194, 165, 150 -> 242.2, 235.1, 230.6

# Перевод в формат для вывода s = np.array(s,dtype=np.uint8) return s

\_4\_log\_c255\_from\_rgb\_img\_1 = get\_log\_from\_rgb(\_source\_img\_1.copy(), 255)

\_4\_log\_c255\_from\_rgb\_img\_2 = get\_log\_from\_rgb(\_source\_img\_2.copy(), 255)

\_4\_log\_c100\_from\_rgb\_img\_1 = get\_log\_from\_rgb(\_source\_img\_1.copy(), 100)

\_4\_log\_c100\_from\_rgb\_img\_2 = get\_log\_from\_rgb(\_source\_img\_2.copy(), 100) cv2.imwrite('\_images/results/\_4\_log\_c255\_from\_rgb\_img\_1.png',

\_4\_log\_c255\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_4\_log\_c255\_from\_rgb\_img\_2.png',

\_4\_log\_c255\_from\_rgb\_img\_2) cv2.imwrite('\_images/results/\_4\_log\_c100\_from\_rgb\_img\_1.png',

\_4\_log\_c100\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_4\_log\_c100\_from\_rgb\_img\_2.png',

\_4\_log\_c100\_from\_rgb\_img\_2)

# \_5\_ Провести степенное преобразование с γ>1, γ<1 def deg\_rgb(r, y):

table = np.array([((i / 255.0) \*\* y) \* 255 for i in np.arange(0, 256)]).astype("uint8")

img\_gamma\_corrected = cv2.LUT(r, table) return img\_gamma\_corrected

\_5\_deg\_more\_1\_from\_rgb\_img\_1 = deg\_rgb(\_source\_img\_1.copy(), 1.5)

\_5\_deg\_more\_1\_from\_rgb\_img\_2 = deg\_rgb(\_source\_img\_2.copy(), 1.5)

\_5\_deg\_less\_1\_from\_rgb\_img\_1 = deg\_rgb(\_source\_img\_1.copy(), 0.5)

\_5\_deg\_less\_1\_from\_rgb\_img\_2 = deg\_rgb(\_source\_img\_2.copy(), 0.5) cv2.imwrite('\_images/results/\_5\_deg\_more\_1\_from\_rgb\_img\_1.png',

\_5\_deg\_more\_1\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_5\_deg\_more\_1\_from\_rgb\_img\_2.png',

\_5\_deg\_more\_1\_from\_rgb\_img\_2) cv2.imwrite('\_images/results/\_5\_deg\_less\_1\_from\_rgb\_img\_1.png',

\_5\_deg\_less\_1\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_5\_deg\_less\_1\_from\_rgb\_img\_2.png',

\_5\_deg\_less\_1\_from\_rgb\_img\_2)

# \_6\_ Провести кусочно-линейное преобразование

def piecewise\_linear\_transformation(img, r1, s1, r2, s2): def piecewise\_linear(x):

if x < r1:

return (s1 / r1) \* x elif r1 <= x < r2:

return ((s2 - s1) / (r2 - r1)) \* (x - r1) + s1 else:

return ((255 - s2) / (255 - r2)) \* (x - r2) + s2

table = np.array([piecewise\_linear(i) for i in range(256)]).astype('uint8') return cv2.LUT(img, table)

\_6\_sl\_from\_rgb\_img\_1 = piecewise\_linear\_transformation(\_source\_img\_1.copy(), 55, 0, 210, 255)

\_6\_sl\_from\_rgb\_img\_2 = piecewise\_linear\_transformation(\_source\_img\_2.copy(), 55, 0, 210, 255)

cv2.imwrite('\_images/results/\_6\_sl\_from\_rgb\_img\_1.png', \_6\_sl\_from\_rgb\_img\_1)

cv2.imwrite('\_images/results/\_6\_sl\_from\_rgb\_img\_2.png', \_6\_sl\_from\_rgb\_img\_2)

# \_7\_ Провести вырезание уровней в изображении (обнуляем каналы, которые не равны вырезаемому)

"""

Пространственная область изображения, это массив пикселей, каждый пиксель обладает определенным значением почернения (светлотой – L).

Значение светлоты L может принимать одно из дискретных значений в интервале [0: L-1]. Число уровней L зависит от числа уровней квантования, заданных при получении или создании изображения. Однако отметим, что число уровней будет рассчитываться как 2n степень n является характеристикой изображения, называемой глубиной цвета и измеряется как число бит на пиксель [bpp].

"""

def cut\_channel\_from\_rgb(\_lab\_color\_img, number\_channel): width, height, channels = \_lab\_color\_img.shape

for i in range(width):

for j in range(height):

color\_rgb = \_lab\_color\_img[i, j] for c in range(channels):

if (number\_channel != c): color\_rgb[c] = 0

\_lab\_color\_img[i, j] = color\_rgb return \_lab\_color\_img

\_source\_rgb\_img = cv2.cvtColor(\_source\_img\_1, cv2.COLOR\_BGR2RGB)

\_7\_channel\_r\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 0)

\_7\_channel\_g\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 1)

\_7\_channel\_b\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 2) cv2.imwrite('\_images/results/\_7\_channel\_r\_from\_rgb\_img\_1.png',

\_7\_channel\_r\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_7\_channel\_g\_from\_rgb\_img\_1.png',

\_7\_channel\_g\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_7\_channel\_b\_from\_rgb\_img\_1.png',

\_7\_channel\_b\_from\_rgb\_img\_1)

def cut\_channel\_from\_rgb(in\_img, number\_level): width, height, channels = in\_img.shape min\_bit = 2 \*\* (number\_level - 1) select\_bit = 2 \*\* number\_level

max\_bit = 2 \*\* (number\_level + 1) for i in range(width):

for j in range(height): r, g, b = in\_img[i, j]

avg\_rgb = (r + g + b) / 3

if (avg\_rgb > min\_bit and avg\_rgb < max\_bit): in\_img[i, j] = r, g, b

else:

in\_img[i, j] = 255, 255, 255 return in\_img

\_7\_bit\_4\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 4)

\_7\_bit\_5\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 5)

\_7\_bit\_6\_from\_rgb\_img\_1 = cut\_channel\_from\_rgb(\_source\_rgb\_img.copy(), 6) cv2.imwrite('\_images/results/\_7\_bit\_1\_from\_rgb\_img\_1.png',

\_7\_bit\_4\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_7\_bit\_5\_from\_rgb\_img\_1.png',

\_7\_bit\_5\_from\_rgb\_img\_1) cv2.imwrite('\_images/results/\_7\_bit\_6\_from\_rgb\_img\_1.png',

\_7\_bit\_6\_from\_rgb\_img\_1)