Дата: 27.03.2023
ФИО: Леонов Владислав Денисович
Группа: 224-322

ПРАКТИЧЕСКАЯ РАБОТА № 2
Применение методов градационной коррекции по переходным кривым

1. Цель работы

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

2. 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 $\gamma>1$, $\gamma<1$
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)
    a. глубина цвета - k, bpp
    b. 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

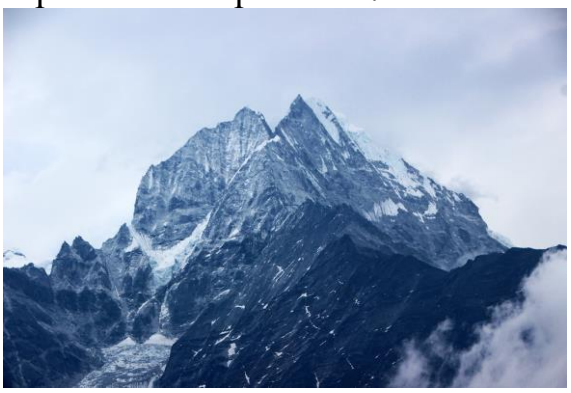
Исходные изображения и все изображения после коррекций выложить на облачное хранилище и приложить ссылку.

3. Исходные данные и программное обеспечение

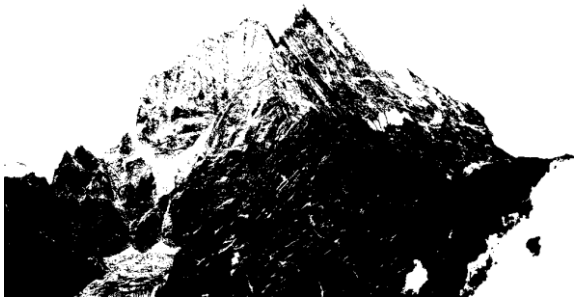
Используемая среда программирования: Visual Studio Code
Используемый язык программирования: Python 3.7.8rc1 64-bit
Используемые библиотеки:
cv2 – https://opencvguide.readthedocs.io/en/latest/opencvpython/basics.html
NumPy - работа с массивами

4. Выполнение работы

1. Подобрать 2 изображения для коррекции
   Подобранные изображения:

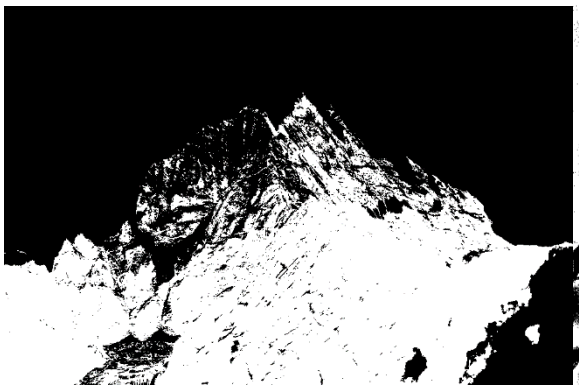



2. Перевести изображения в черно-белое
   Изображения в черно-белом на основе яркости пространства LAB:




   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.

3. Преобразовать изображения в негатив
   Изображения в негативе на основе яркости пространства LAB:

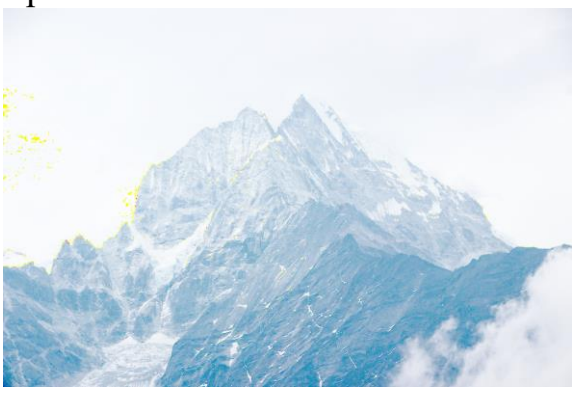



Изображения в негативе на основе инвертированных пикселей пространства RGB:

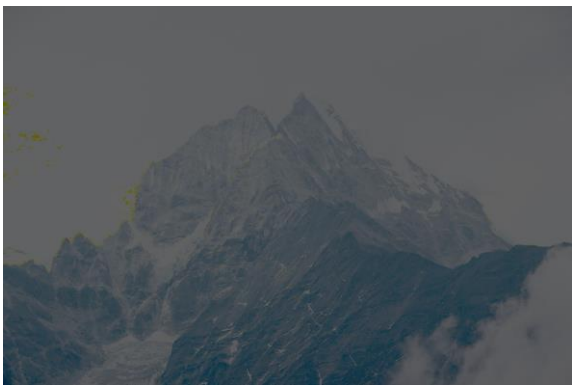

Для преобразования в негатив, используется функция s=255-r, где s – результат преобразования, r – исходное изображение, в данной функции происходит инвертирование пикселей на изображении, если они были белыми, то станут черными.

4. Провести логарифмическое преобразование

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



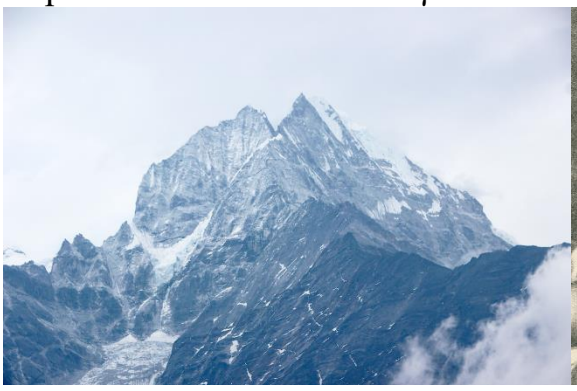

Изображения после логарифмического преобразования на основе пространства RGB с значением c = 100:




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.

5. Провести степенное преобразование с $\gamma>1$, $\gamma<1$
Изображения после степенного преобразования на основе пространства RGB с гаммой $\gamma = 1.5$:


Изображения после степенного преобразования на основе пространства RGB с гаммой $\gamma = 0.5$:




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.

6. 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
```

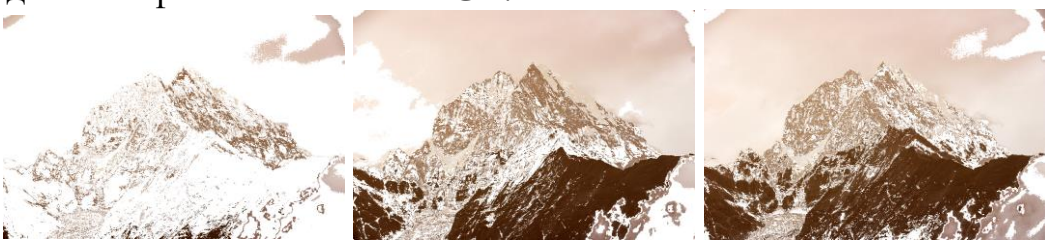
   Параметры: r – входное изображение (пиксели), r1 – параметр первой границы, например, до 55, s1 – результирующий параметр первой границы, например, 0, означает, что результирующие пиксели станут 0, r2 – параметр второй границы, s2 – результирующий параметр второй границы.

7. 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/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)
```