

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806: «Вычислительная математика и программирование»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №2

По курсу: «Цифровая обработка изображений»

Тема: «Методы поэлементной обработки изображений»

Студент: Чернышев Д.В.

Группа: М8О-107М-22

Вариант: 8

Преподаватель: Гаврилов К.Ю.

Москва

2023

СОДЕРЖАНИЕ

1	Задание к лабораторной работе №2.....	3
2	Выполнение лабораторной работы №2	6
3	Выводы	20

1 Задание к лабораторной работе №2

Часть 1

Исходное изображение "Img2_08_1.jpg" представляет собой черно-белую фотографию. Создайте имитацию неравномерной засветки изображения, равномерно увеличивающейся от центра к краям (линии равной яркости – окружности). Затем, используя функцию ‘imflatfield’, восстановите исходное изображение. При восстановлении изображения подберите оптимальные значения параметров функции ‘imflatfield’, обеспечивающие наивысшую равномерность яркости по всей поверхности изображения. При необходимости используйте функцию эквализации гистограммы.



Рис. 1: Изображение для первого задания.

Часть 2

Исходное изображение представлено в виде двумерной матрицы, записанной в формате Matlab в файле Z2_08_2.mat. Определите динамический диапазон яркостей изображения и подберите вид нелинейного преобразования, при котором можно будет различить все детали изображения как в центре, так

и на краях. Обеспечьте возможность изменения крутизны нелинейности преобразования, определяющей соотношения яркостей промежуточных тонов. Использование для улучшения яркости и контраста изображений следующих функций:

- *imadjust* – реализует гамма-коррекцию,
- *histeq* – реализует метод эквализации гистограммы,
- *imlocalbrighten* – выполняет автоматическое выравнивания яркости темных областей,
- *imcontrast* – выполняет ручную подстройку яркости и контраста.

Часть 3

Дано изображение *Img2_08_3.jpg*. Постройте гистограмму яркости заданного изображения.

- а) Используйте функцию *imadjust* для коррекции контраста.
- б) Постройте гистограмму яркости скорректированного изображения.
- в) Сравните гистограммы яркостей обоих изображений и сделайте выводы.
- г) Используйте функцию *histeq* для эквализации гистограммы.

Постройте гистограмму яркости после эквализации. Сравните гистограммы яркостей, полученные в результате использования функций *imadjust* и *histeq*. Используйте функцию *imlocalbrighten* для улучшения контраста.

Постройте гистограмму скорректированного изображения.



Рис. 2: *Img2_08_3.jpg*



Рис. 3: *Img2_08_3.jpg* - оригинал

2 Выполнение лабораторной работы №2

Часть 1

Для начала подгружим необходимые модули для работы

```
1 from PIL import Image, ImageDraw, ImageStat  
2 import numpy as np  
3 import cv2  
4 import bisect  
5 from numba import jit  
6 import os  
7 import IPython.display #import Image  
8 import matplotlib.pyplot as plt
```

Напишем аналог функции *imflatfield*, а также функции зашумления исходного изображения

```
1 def imflatfield(I, sigma):
2     """Python equivalent imflatfield implementation
3         I format must be BGR and type of I must be uint8"""
4     A = I.astype(np.float32) / 255 # A = im2single(I);
5     Ihsv = cv2.cvtColor(A, cv2.COLOR_BGR2HSV)
6     A = Ihsv[:, :, 2] # A = Ihsv(:,:,3);
7
8     filterSize = int(2*np.ceil(2*sigma) + 1)
9     shading = cv2.GaussianBlur(
10         A,
11         (filterSize, filterSize),
12         sigma, borderType=cv2.BORDER_REFLECT
13     )
14     meanVal = np.mean(A)
15     shading = np.maximum(shading, 1e-6)
16     B = A*meanVal / shading
17     Ihsv[:, :, 2] = B
18     B = cv2.cvtColor(Ihsv, cv2.COLOR_HSV2BGR)
19     B = np.round(np.clip(B*255, 0, 255)).astype(np.uint8)
20     return B
21
22 def make_noisy_image(
23     OUTPUT_DIR:str,
24     FILE:str,
25     loc:float=0,
26     scale:float=1)-> str:
27     img = cv2.imread(FILE)
28     # Получение ширины, высоты и количество каналов изображения
29     h, w, _ = img.shape
30     # Создание сетки из точек вокруг центра изображения
31     x, y = np.meshgrid(
```

```
32         np.linspace(-1, 1, w),
33         np.linspace(-1, 1, h)
34     )
35
36     # Вычисление расстояния каждой точки сетки
37     # от центра изображения
38     radius = np.sqrt(x**2 + y**2)
39
40
41     # Нормализация радиуса для создания градиента яркости
42     radius = (radius - radius.min()) / (
43         radius.max() - radius.min()
44     )
45
46     np.random.normal()
47
48     # Создание маски, которая определяет
49     # область имитации неравномерной засветки
50     mask = np.random.normal(loc=0,scale=1,size=(h, w))
51     mask[radius > 0.95] = 0
52
53
54     # Накладывание маски на изображение и сохранение результата
55     result = cv2.merge(
56         [img[:, :, 0]*mask,
57          img[:, :, 1]*mask,
58          img[:, :, 2]*mask]
59     )
60
61     cv2.imwrite(
62         os.path.join(
63             OUTPUT_DIR,f"NoiseImage(loc={loc}-scale={scale}).jpg"
64             ),
65         result
66     )
67
68     return os.path.join(
69         OUTPUT_DIR,f"NoiseImage(loc={loc}-scale={scale}).jpg"
70     )
```

```
65 def make_dark_spot(
66     OUTPUT_DIR:str,
67     FILE:str):
68     # Загружаем изображение в оттенках серого
69     name = os.path.split(FILE)[-1]
70     img_gray = cv2.imread(FILE, cv2.IMREAD_GRAYSCALE)
71
72     # Получаем размеры изображения
73     height, width = img_gray.shape
74
75     # Вычисляем центр изображения
76     center_x = int(width/2)
77     center_y = int(height/2)
78
79     # Создаем новое черное изображение
80     result = np.zeros((height, width), np.uint8)
81
82     # Проходимся по всем пикселям изображения
83     for x in range(width):
84         for y in range(height):
85             # Вычисляем расстояние до центра
86             distance = (
87                 (x - center_x) ** 2
88                 + (y - center_y) ** 2
89                 ) ** 0.5
90             radius = (
91                 (height - center_x) ** 2
92                 + (width - center_y) ** 2
93                 ) ** 0.5
94             # Нормализуем расстояние, получаем
95             # значение от 0 до 1
96             norm_distance = distance / radius
97
```

```
98 # Вычисляем яркость пикселя на основе
99 # обратно пропорционального значения
100 # нормализованного расстояния
101     if norm_distance !=0:
102         brightness = int(255 * (norm_distance))
103     else:
104         brightness = 1e-10
105
106     # Применяем яркость к пикслю изображения
107     result[y, x] = img_gray[y, x] * brightness // 255
108
109 # Показываем результат и сохраняем его
110 cv2.imwrite(
111     os.path.join(OUTPUT_DIR,f"WithDarkSpot_{name}"),
112     result
113 )
114 return os.path.join(OUTPUT_DIR,f"WithDarkSpot_{name}")
```

В главной части кода

- 1) Создание изображения с неравномерной темной засветкой увеличивающейся от центра к краям.
- 2) В цикле от 30 до 300 с шагом 30 применяют к "испорченному" изображению функцию *imflatfield*

```
1 img = cv2.imread(make_dark_spot(OUTPUT_PIC,SOURCE_FILE), 0)
2 img = cv2.equalizeHist(img, 0)
3 OUTPUT_ImFlatField = os.path.join(OUTPUT_PIC , "ImFlatField")
4 if not os.path.exists(OUTPUT_ImFlatField):
5     os.makedirs(OUTPUT_ImFlatField)
6 HISTED_file = os.path.join(OUTPUT_ImFlatField ,f"HIST_{pic}")
7 cv2.imwrite(HISTED_file, img)
8
9 I = cv2.imread(HISTED_file)
10 for sigma in range(30,300,30):
11     flatField_file = os.path.join(
12         OUTPUT_ImFlatField ,f"(sigma,{sigma})_{pic}"
13     )
14     test = imflatfield(I, sigma)
15     cv2.imwrite(flatField_file, test)
```



Рис. 4: Результаты первой части задания

Часть 2

Для выполнения поставленной необходимо реализовать функцию *imadjust*

```

1 @jit
2 def imadjust(src, tol=1, vin=[0,255], vout=(0,255)):
3     # src : input one-layer image (numpy array)
4     # tol : tolerance, from 0 to 100.
5     # vin : src image bounds
6     # vout : dst image bounds
7     # return : output img
8
9     assert len(src.shape) == 2 , 'Input image should be 2-dims'
10
11    tol = max(0, min(100, tol))
12

```

```

13     if tol > 0:
14
15         # Compute in and out limits
16
17         # Histogram
18
19         hist = np.histogram(
20
21             src,
22
23             bins=list(range(256)),
24
25             range=(0,255)
26
27         )[0]
28
29
30         # Cumulative histogram
31
32         cum = hist.copy()
33
34         for i in range(1, 256): cum[i] = cum[i - 1] + hist[i]
35
36
37         # Compute bounds
38
39         total = src.shape[0] * src.shape[1]
40
41         low_bound = total * tol / 100
42
43         upp_bound = total * (100 - tol) / 100
44
45         vin[0] = bisect.bisect_left(cum, low_bound)
46
47         vin[1] = bisect.bisect_left(cum, upp_bound)
48
49
50
51         # Stretching
52
53         scale = (vout[1] - vout[0]) / (vin[1] - vin[0])
54
55         vs = src-vin[0]
56
57         vs[src<vin[0]]=0
58
59         vd = vs*scale+0.5 + vout[0]
60
61         vd[vd>vout[1]] = vout[1]
62
63         dst = vd
64
65
66     return dst

```

Визуализируем изображение из матрицы:

```

1 from matplotlib.pyplot import imshow
2

```

```
3 MAT = MAT_RAW_FIX.reshape(*MAT_RAW_FIX.shape)
4 MAT = MAT.astype(np.uint8)
5 MAT = np.stack((MAT,) * 3, axis=-1)
6
7 image_raw = Image.fromarray(MAT)
8 image_raw.save(os.path.join(OUTPUT, "raw_image.png"))
9 %matplotlib inline
10 imshow(image_raw)
```

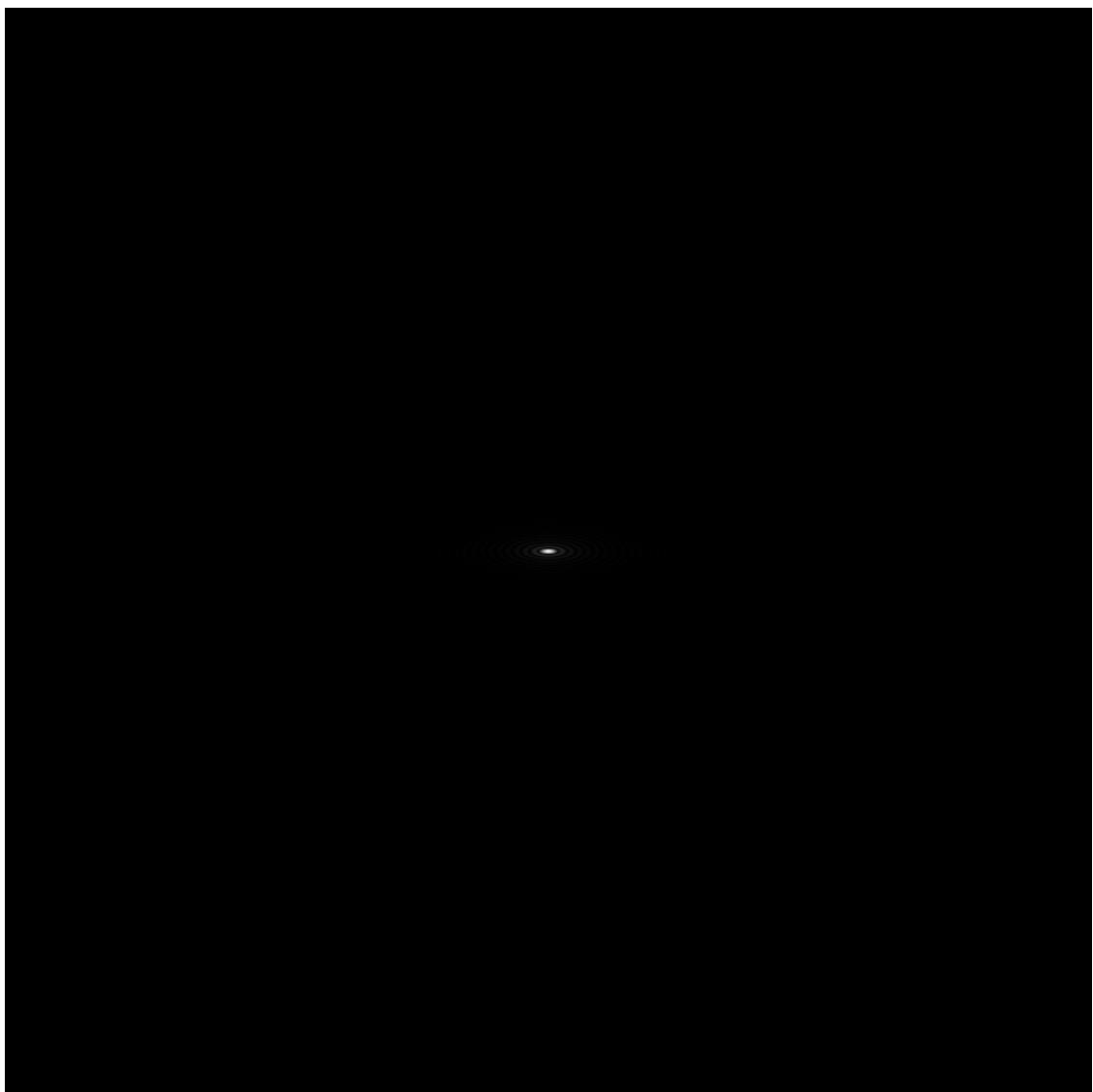


Рис. 5: Визуализация данных матрицы Z2_08_2.mat.

Применим теперь функцию *imadjust* к нашему изображению и поэкспериментируем с разными параметрами

```
1 for tol_i in range(5,100,10):
2     tol = tol_i/100
3     image = imadjust(MAT_RAW_FIX,tol=tol)
4     MAT = image.reshape(*image.shape).astype(np.uint8)
5     MAT = np.stack((MAT,) * 3, axis=-1)
6     image_data = Image.fromarray(MAT)
7     image_data.save(
8         os.path.join(OUTPUT,f"Tol({tol})_image.png")
9     )
```

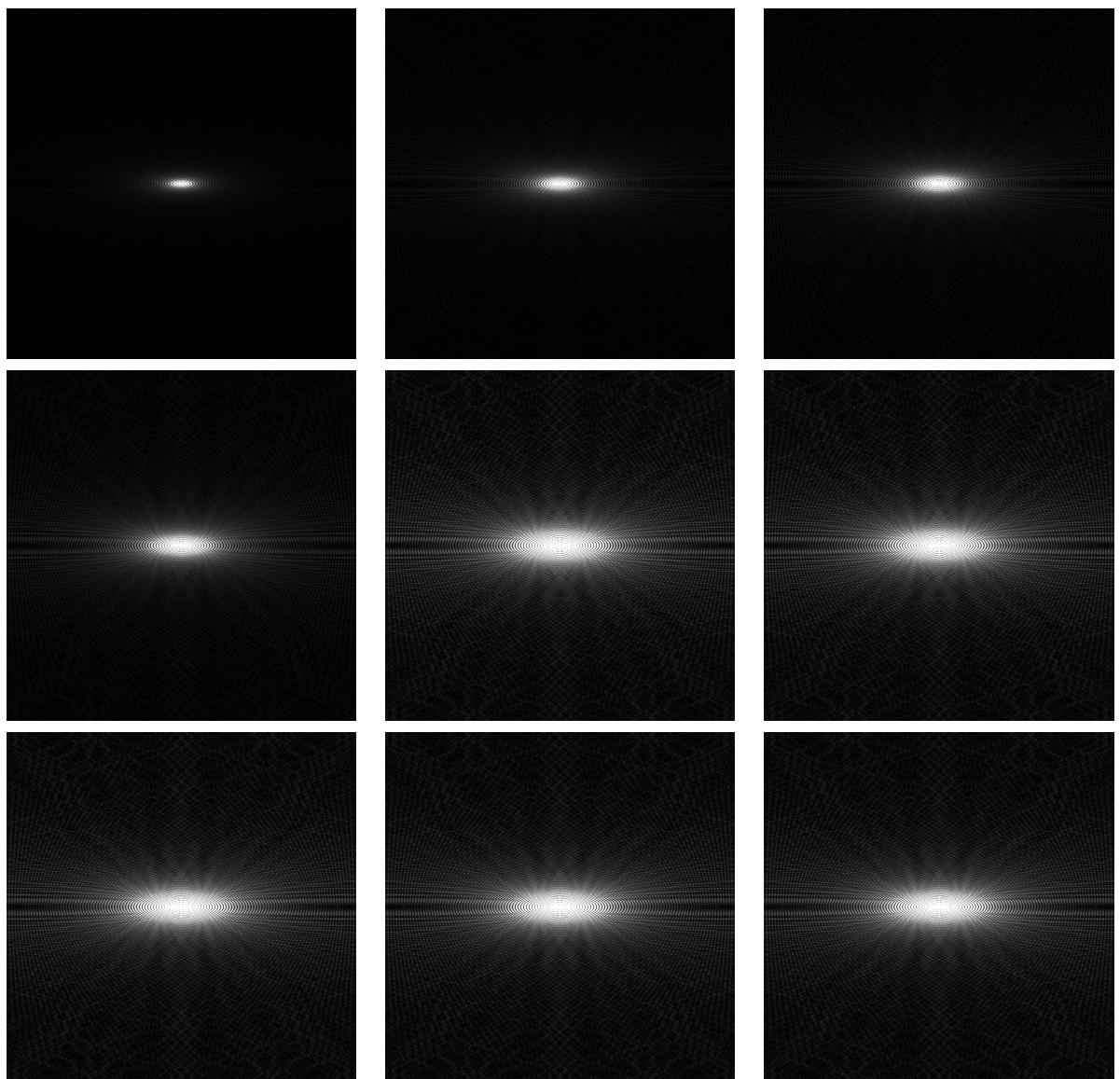


Рис. 6: Результаты второй части задания

Часть 3

Функция построения гистограммы изображения:

```
1 def color_hist(filename):
2     img = np.asarray(
3         Image.open(filename).convert("L")
4         ).reshape(-1,1)
5     plt.hist(img, bins=128)
6     plt.show()
```

Исходное изображение и его гистограмма:



Рис. 7: Исходное изображение.

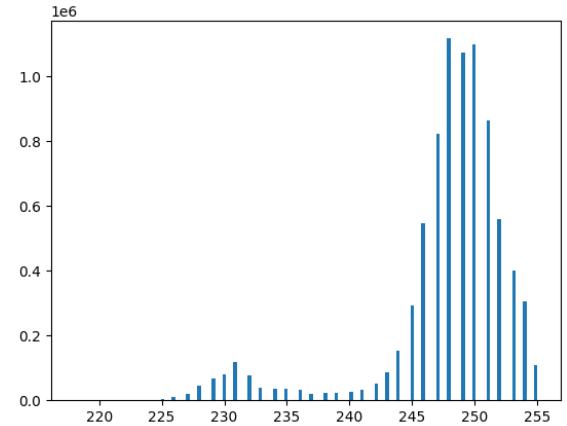


Рис. 8: Гистограмма исходного изображения.

Применим функцию *imadjust*:

```
1 tol = 1/2+12.5/2
2 saved_image_path = os.path.join(
3     OUTPUT_PIC_3,f"IMADJUST_(tol={tol})_{pic}"
4 )
5 image = Image.open(SOURCE_FILE_3)
6 arr = np.asarray(image)
7 arr2=imadjust(arr,tol=tol)
8 new_im = Image.fromarray(arr2)
9 if new_im.mode != 'RGB':
10     new_im = new_im.convert('RGB')
11 new_im.save(saved_image_path)
```



Рис. 9: Применение *imadjust* с аргументом функции 6.75

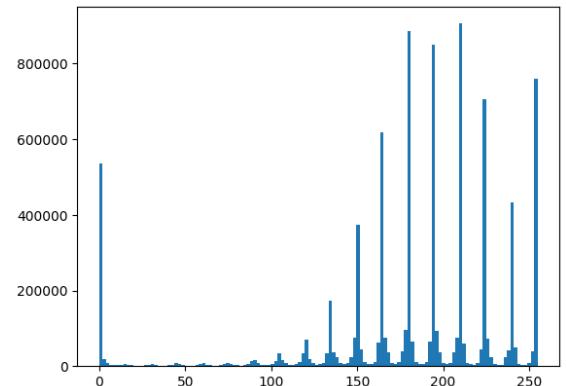


Рис. 10: Гистограмма

Применение эквализации:

```
1 img = cv2.imread(saved_image_path, 0)

2

3 # creating a Histograms Equalization
4 # of a image using cv2.equalizeHist()
5 equ = cv2.equalizeHist(img)

6

7 saved_image_path_equalize = os.path.join(
8     OUTPUT_PIC_3,f"EQUALIZED_{pic}"
9 )

10

11 imgplot = plt.imshow(equ)
12 plt.show()
13 cv2.imwrite(saved_image_path_equalize, equ)
14 cv2.waitKey(0)
```



Рис. 11: Применение эквализации

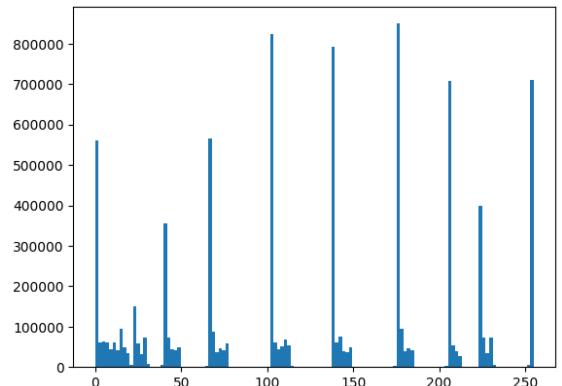


Рис. 12: Гистограмма

По заданию необходимо было применить функцию *imlocalbrighten*

```
1 def my_imlocalbrighten(
2     FILE:str=saved_image_path_equlize,
3     val:int=45,
4     OUTPUT:str=OUTPUT_PIC_3
5 ):
6
7     name = os.path.split(FILE)[-1]
8     # Open the image
9     im = Image.open(FILE)
10
11    # Convert to HSV colourspace
12    # and split channels for ease of separate processing
13    H, S, V = im.convert('HSV').split()
14
15    # Increase the brightness, or Value channel
16    # Change 30 to 50 for bigger effect,
17    # or 10 for smaller effect
18    newV = V.point(lambda i: i + int(val*(255-i)/255))
19
20    # Recombine channels and convert back to RGB
21    res = Image.merge(
22        mode="HSV",
23        bands=(H,S,newV)).convert('RGB')
24
25    res.save(
26        os.path.join(OUTPUT,f"imlocalbrighten_{name}")
27    )
28    return os.path.join(OUTPUT,f"imlocalbrighten_{name}")
```

В результате применения данной функции будем иметь:



Рис. 13: Применение функции
imlocalbrighten

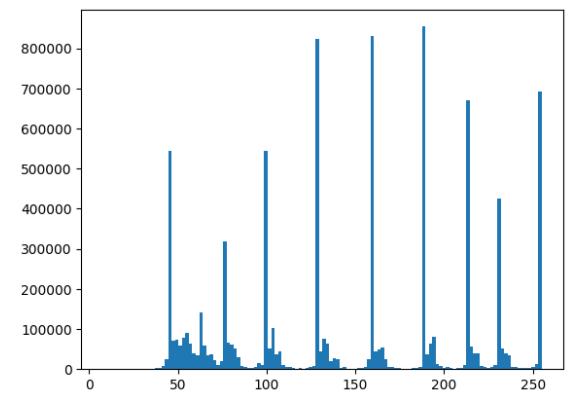


Рис. 14: Гистограмма

3 Выводы

В ходе выполнения данной лабораторной работы были обработаны различные изображения с разным уровнем зашумленности с применением таких функций

- *imadjust*
- *histeq*
- *imlocalbrighten*
- *imcontrast*.

Все откорректированные изображения были проанализированы, по результатам также были составлены гистограммы яркости.