

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(УНИВЕРСИТЕТ ИТМО)

Факультет «Систем управлени и робототехники»

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

По дисциплине «Техническое зрение»
на тему: «Гистограммы, профили и проекции»

Студент:
Охрименко Ева ИСУ 409290

Проверил:
Шаветов Сергей Васильевич

г. Санкт-Петербург
2024

Содержание

1 Task. Гистограммы	3
1.1 Функция <code>plot_histograms</code> . Построение гистограмм	3
1.1.1 Код	3
1.1.2 Объяснение	3
1.1.3 Итог	3
1.1.4 Выход	3
1.2 Функция <code>uniform_transform</code> . Равномерное преобразование	4
1.2.1 Код	4
1.2.2 Объяснение	4
1.2.3 Итог	4
1.2.4 Выход	5
1.3 Функция <code>arithmetic_operations</code> . Арифметические операции	5
1.3.1 Код	5
1.3.2 Объяснение	5
1.3.3 Итог	6
1.3.4 Выход	6
1.4 Функция <code>dynamic_range_stretching</code> . Растворение динамического диапазона	6
1.4.1 Код	6
1.4.2 Объяснение	7
1.4.3 Итог	7
1.4.4 Выход	7
1.5 Функция <code>exponential_transform</code> . Экспоненциальное преобразование	8
1.5.1 Код	8
1.5.2 Объяснение	8
1.5.3 Итог	8
1.5.4 Выход	9
1.6 Функция <code>rayleigh_transform</code> . Преобразование по закону Рэлея	9
1.6.1 Код	9
1.6.2 Объяснение	10
1.6.3 Итог	10
1.6.4 Выход	10
1.7 Функция <code>power_law_transform</code> . Степенное преобразование	10
1.7.1 Код	11
1.7.2 Объяснение	11
1.7.3 Итог	11
1.7.4 Выход	11
1.8 Функция <code>hyperbolic_transform</code> . Гиперболическое преобразование	12
1.8.1 Код	12
1.8.2 Объяснение	12
1.8.3 Итог	12
1.8.4 Выход	13
1.9 Вывод	13
1.9.1 Арифметические операции	13
1.9.2 Растворение динамического диапазона	13
1.9.3 Равномерное преобразование	13
1.9.4 Экспоненциальное преобразование	14
1.9.5 Преобразование по закону Рэлея	14
1.9.6 Преобразование по закону степени 2/3	14

1.9.7	Гиперболическое преобразование	14
2	Task. Профили	14
2.1	Функция <code>plot_brightness_profile</code> . Построение профиля яркости	14
2.1.1	Код	14
2.1.2	Объяснение	15
2.1.3	Итог	15
2.1.4	Выход	15
3	Task. Проекции	16
3.1	Функция <code>render_projections</code> . Построение проекций изображения	16
3.1.1	Код	16
3.1.2	Объяснение	16
3.1.3	Итог	17
3.1.4	Выход	17
4	Вывод	17

1 Task. Гистограммы

1.1 Функция plot_histograms. Построение гистограмм

Функция строит гистограммы для исходного изображения.

1.1.1 Код

```
def plot_histograms(original, transformed, title):
    colors = ('b', 'g', 'r')
    plt.figure(figsize=(14, 6))

    plt.subplot(2, 2, 1)
    for i, color in enumerate(colors):
        hist = cv2.calcHist([original], [i], None, [256], [0, 256])
        hist = hist / (original.shape[0] * original.shape[1])
        plt.plot(hist, color=color)
        plt.xlim([0, 256])
    plt.title("Гистограмма исходного изображения")
    plt.xlabel('Интенсивность пикселей')
    plt.ylabel('Плотность пикселей')
```

1.1.2 Объяснение

Функция принимает исходное и преобразованное изображения, а также заголовок. Для каждого канала (B, G, R) вычисляется гистограмма с помощью `cv2.calcHist`. Гистограмма нормируется на общее количество пикселей, после чего строится график. Добавляются подписи осей и заголовок.

1.1.3 Итог

На вход подаются исходное и преобразованное изображения. На выходе получается график гистограммы исходного изображения.

1.1.4 Выход



Рис. 1: Исходное изображение

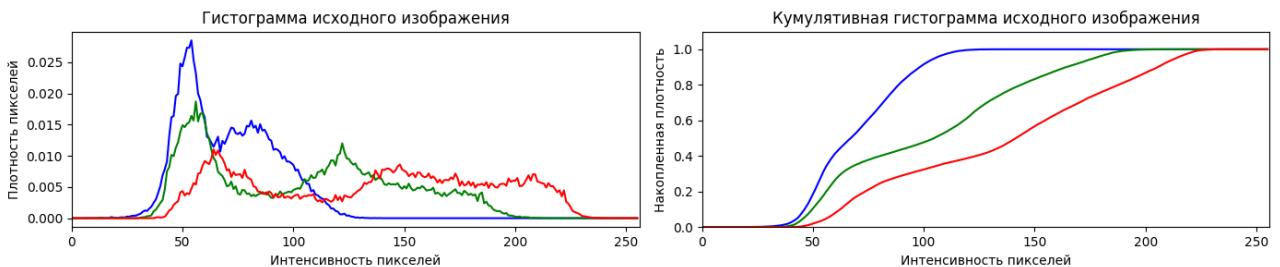


Рис. 2: Гистограмма исходного изображения

1.2 Функция uniform_transform. Равномерное преобразование

Функция растягивает интенсивности изображения на весь диапазон $[0, 1]$ или $[0, 255]$.

1.2.1 Код

```
def uniform_transform(I):
    Inew = I.astype(np.float32) / 255 if I.dtype == np.uint8 else I
    Imin, Imax = Inew.min(), Inew.max()
    Inew = (Inew - Imin) / (Imax - Imin)
    return (Inew * 255).clip(0, 255).astype(np.uint8) if I.dtype == np.uint8 else Inew
```

1.2.2 Объяснение

Функция принимает изображение и выполняет растяжение динамического диапазона. Если изображение в формате `uint8`, оно нормализуется в диапазон $[0, 1]$. Затем находятся минимальное (`Imin`) и максимальное (`Imax`) значения интенсивности. Интенсивности растягиваются на весь диапазон $[0, 1]$ с помощью формулы:

$$I_{\text{new}} = \frac{I_{\text{new}} - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}}.$$

Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.2.3 Итог

На вход подается изображение. На выходе получается изображение с улучшенным контрастом, где интенсивности растянуты на весь доступный диапазон.

1.2.4 Выход



Рис. 3: Полученное изображение после равномерного преобразования

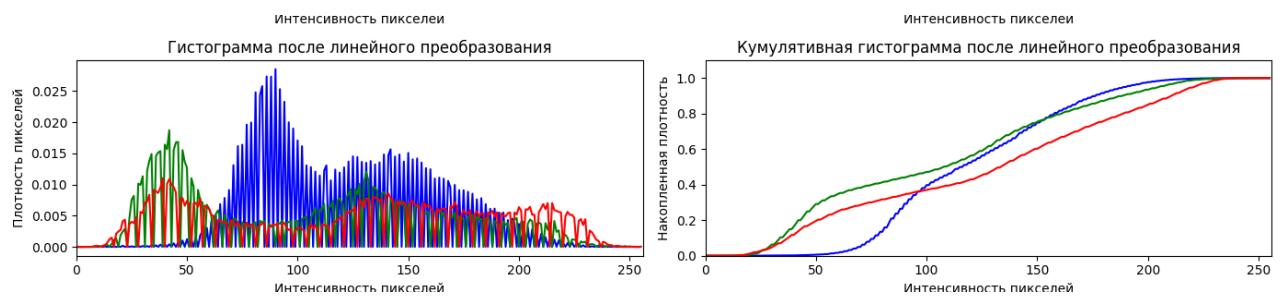


Рис. 4: Гистограмма после равномерного преобразования

1.3 Функция arithmetic_operations. Арифметические операции

Функция увеличивает яркость изображения на заданное значение.

1.3.1 Код

```
def arithmetic_operations(I, value=50):
    Inew = I.astype(np.float32) + value / 255
    Inew = np.clip(Inew, 0, 1)
    if I.dtype == np.uint8:
        Inew = (255 * Inew).clip(0, 255).astype(np.uint8)
    return Inew
```

1.3.2 Объяснение

Функция принимает изображение и значение `value`, на которое увеличивается яркость. Изображение преобразуется в формат `float32`, и к его интенсивностям добавляется значение `value / 255`. Результат обрезается до диапазона $[0, 1]$. Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.3.3 Итог

На вход подается изображение и значение `value`. На выходе получается изображение с увеличенной яркостью. Если исходное изображение было в формате `uint8`, результат также будет в этом формате.

1.3.4 Выход



Рис. 5: Полученное изображение после арифметической операции

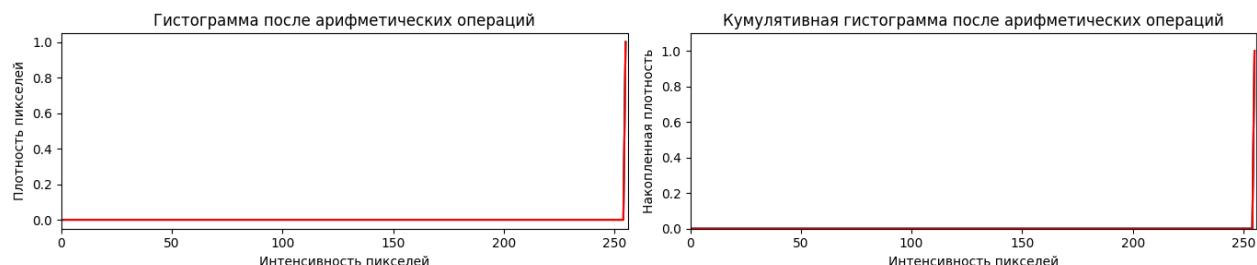


Рис. 6: Гистограмма после арифметической операции

1.4 Функция dynamic_range_stretching. Растяжение динамического диапазона

Функция выполняет растяжение динамического диапазона изображения, улучшая его контраст.

1.4.1 Код

```
def dynamic_range_stretching(I):
    if I.dtype == np.uint8:
        Inew = I.astype(np.float32) / 255
    else:
        Inew = I

    I_BGR = cv2.split(Inew)
    Inew_BGR = []
```

```

for layer in I_BGR:
    Imin = layer.min()
    Imax = layer.max()
    Inew_layer = (layer - Imin) / (Imax - Imin)
    Inew_BGR.append(Inew_layer)

Inew = cv2.merge(Inew_BGR)
if I.dtype == np.uint8:
    Inew = (255 * Inew).clip(0, 255).astype(np.uint8)

return Inew

```

1.4.2 Объяснение

Функция принимает изображение и выполняет растяжение динамического диапазона для каждого канала (B, G, R). Если изображение в формате `uint8`, оно нормализуется в диапазон [0, 1]. Для каждого канала находятся минимальное (`Imin`) и максимальное (`Imax`) значения интенсивности, после чего интенсивности растягиваются на весь диапазон [0, 1] с помощью формулы:

$$I_{\text{new_layer}} = \frac{\text{layer} - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}}.$$

Преобразованные каналы объединяются обратно в одно изображение.

1.4.3 Итог

На вход подается изображение. На выходе получается изображение с улучшенным контрастом, где интенсивности каждого канала растянуты на весь диапазон [0, 1].

1.4.4 Выход



Рис. 7: Полученное изображение

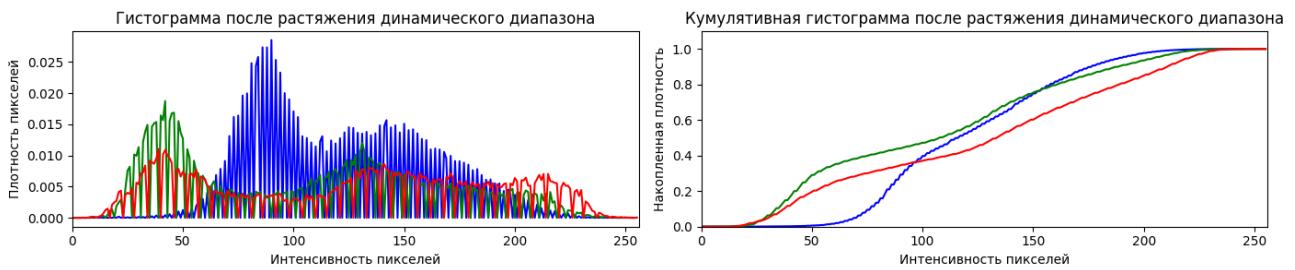


Рис. 8: Гистограмма после растяжения диапазона

1.5 Функция `exponential_transform`. Экспоненциальное преобразование

Функция применяет экспоненциальное преобразование к изображению с параметром α .

1.5.1 Код

```
def exponential_transform(I, alpha=1.0):
    if I.dtype == np.uint8:
        Inew = I.astype(np.float32) / 255
    else:
        Inew = I

    Imin = Inew.min()

    Inew = Imin - (1 / alpha) * np.log(1 - Inew)

    Inew = np.clip(Inew, 0, 1)

    if I.dtype == np.uint8:
        Inew = (255 * Inew).clip(0, 255).astype(np.uint8)

    return Inew
```

1.5.2 Объяснение

Функция принимает изображение и параметр α , который определяет форму экспоненциального преобразования. Если изображение в формате `uint8`, оно нормализуется в диапазон $[0, 1]$. Затем находится минимальное значение интенсивности (`Imin`). К изображению применяется экспоненциальное преобразование:

$$Inew = Imin - \frac{1}{\alpha} \cdot \ln(1 - Inew).$$

Результат обрезается до диапазона $[0, 1]$. Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.5.3 Итог

На вход подается изображение и параметр α . На выходе получается изображение, к которому применено экспоненциальное преобразование. Если исходное изображение было в формате `uint8`, результат также будет в этом формате.

1.5.4 Выход



Рис. 9: Полученное изображение после экспоненциального преобразования

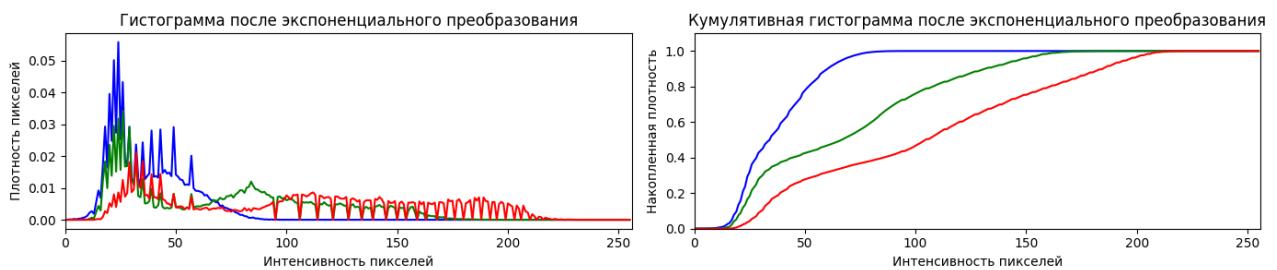


Рис. 10: Гистограмма после экспоненциального преобразования

1.6 Функция rayleigh_transform. Преобразование по закону Рэлея

Функция применяет преобразование интенсивности изображения по закону Рэлея с параметром α .

1.6.1 Код

```
def rayleigh_transform(I, alpha=1.0):
    if I.dtype == np.uint8:
        Inew = I.astype(np.float32) / 255
    else:
        Inew = I

    Imin = Inew.min()

    Inew = Imin + np.sqrt(2 * alpha**2 * np.log(1 / (1 - Inew)))

    Inew = np.clip(Inew, 0, 1)

    if I.dtype == np.uint8:
        Inew = (255 * Inew).clip(0, 255).astype(np.uint8)

    return Inew
```

1.6.2 Объяснение

Функция принимает изображение и параметр α , который определяет форму распределения Рэлея. Если изображение в формате `uint8`, оно нормализуется в диапазон $[0, 1]$. Затем находится минимальное значение интенсивности (I_{min}). К изображению применяется преобразование по закону Рэлея:

$$I_{new} = I_{min} + \sqrt{2\alpha^2 \ln \left(\frac{1}{1 - I_{new}} \right)}.$$

Результат обрезается до диапазона $[0, 1]$. Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.6.3 Итог

На вход подается изображение и параметр α . На выходе получается изображение, к которому применено преобразование по закону Рэлея. Если исходное изображение было в формате `uint8`, результат также будет в этом формате.

1.6.4 Выход



Рис. 11: Полученное изображение после преобразования Рэлея

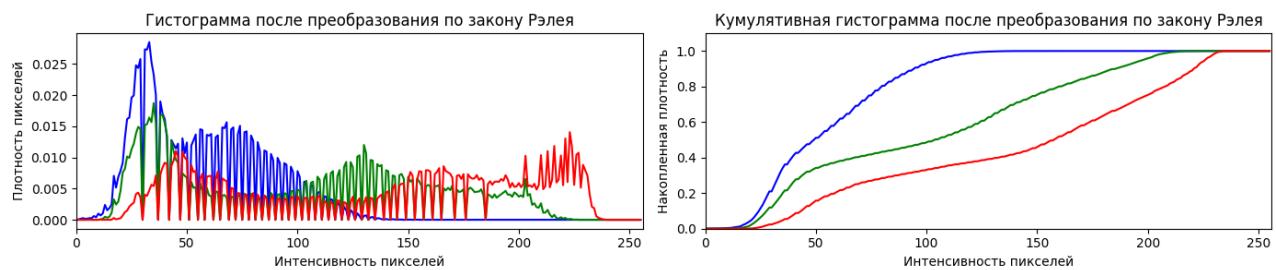


Рис. 12: Гистограмма после преобразования Рэлея

1.7 Функция `power_law_transform`. Степенное преобразование

Функция применяет степенное преобразование к изображению с показателем степени $\frac{2}{3}$.

1.7.1 Код

```
def power_law_transform(I):
    if I.dtype == np.uint8:
        Inew = I.astype(np.float32) / 255
    else:
        Inew = I

    Inew = np.power(Inew, 2/3)
    Inew = np.clip(Inew, 0, 1)

    if I.dtype == np.uint8:
        Inew = (255 * Inew).clip(0, 255).astype(np.uint8)

    return Inew
```

1.7.2 Объяснение

Функция принимает изображение и применяет к нему степенное преобразование с показателем степени $\frac{2}{3}$. Если изображение в формате `uint8`, оно нормализуется в диапазон [0, 1]. Затем к изображению применяется преобразование:

$$I_{\text{new}} = I_{\text{new}}^{\frac{2}{3}}.$$

Результат обрезается до диапазона [0, 1]. Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.7.3 Итог

На вход подается изображение. На выходе получается изображение, к которому применено степенное преобразование с показателем степени $\frac{2}{3}$. Если исходное изображение было в формате `uint8`, результат также будет в этом формате.

1.7.4 Выход



Рис. 13: Полученное изображение после степенного преобразования

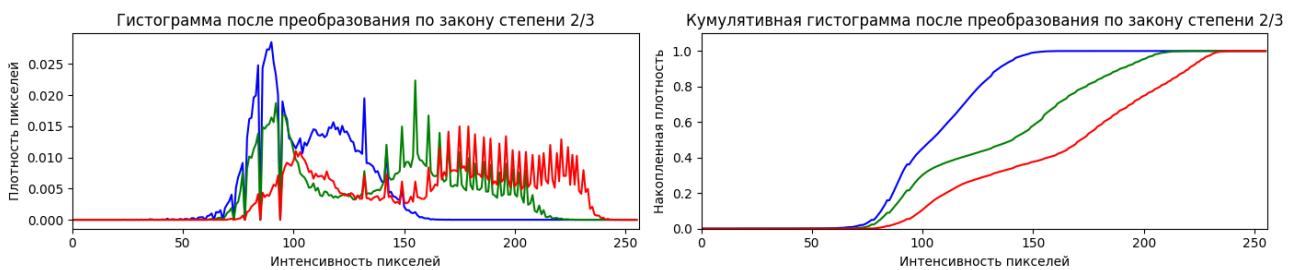


Рис. 14: Гистограмма после степенного преобразования

1.8 Функция `hyperbolic_transform`. Гиперболическое преобразование

Функция применяет гиперболическое преобразование к изображению с параметром α .

1.8.1 Код

```
def hyperbolic_transform(I, alpha=2.0):
    if I.dtype == np.uint8:
        P_I = I.astype(np.float32) / 255
    else:
        P_I = I

    Inew = np.power(alpha, P_I)

    Inew = np.clip(Inew, 0, 1)

    if I.dtype == np.uint8:
        Inew = (255 * Inew).clip(0, 255).astype(np.uint8)

    return Inew
```

1.8.2 Объяснение

Функция принимает изображение и параметр α , который определяет основание гиперболического преобразования. Если изображение в формате `uint8`, оно нормализуется в диапазон $[0, 1]$. Затем к изображению применяется гиперболическое преобразование:

$$I_{\text{new}} = \alpha^{P_I}.$$

Результат обрезается до диапазона $[0, 1]$. Если исходное изображение было в формате `uint8`, результат преобразуется обратно в этот формат.

1.8.3 Итог

На вход подается изображение и параметр α . На выходе получается изображение, к которому применено гиперболическое преобразование. Если исходное изображение было в формате `uint8`, результат также будет в этом формате.

1.8.4 Выход



Рис. 15: Полученное изображение после гиперболического преобразования

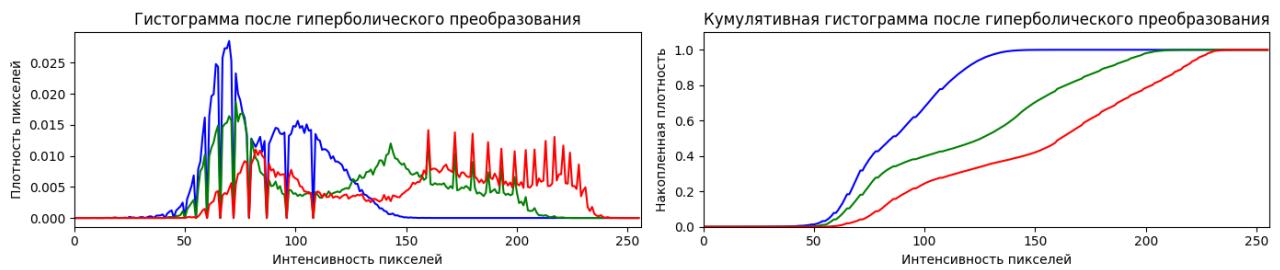


Рис. 16: Гистограмма после гиперболического преобразования

1.9 Вывод

В этом задании я разобралась с разными методами улучшения изображений и реализовала их на практике. Каждый метод я протестировала на слабоконтрастной картинке , а результаты визуализировала с помощью гистограмм. Вот что у меня получилось:

1.9.1 Арифметические операции

С помощью арифметических операций можно легко увеличить яркость изображения. Это особенно полезно, если картинка слишком темная. Метод простой и быстрый, поэтому его удобно использовать для базовой обработки.

1.9.2 Растворение динамического диапазона

Этот метод отлично подходит для изображений, где яркости сосредоточены в узком диапазоне. Он растягивает их на весь доступный диапазон, делая картинку более выразительной.

1.9.3 Равномерное преобразование

Равномерное преобразование помогает растянуть яркости на весь диапазон, что особенно полезно для картинок с плохим контрастом. В результате изображение становится более детализированным и приятным для глаз.

1.9.4 Экспоненциальное преобразование

Экспоненциальное преобразование позволяет гибко настраивать контраст с помощью параметра α . Этот метод особенно полезен, если на изображении есть пересвеченные или слишком темные участки.

1.9.5 Преобразование по закону Рэлея

Этот метод хорошо справляется с изображениями, где есть шумы или неравномерное освещение. Параметр α помогает адаптировать преобразование под конкретные задачи, улучшая видимость деталей в сложных условиях.

1.9.6 Преобразование по закону степени 2/3

Степенное преобразование с показателем $\frac{2}{3}$ улучшает видимость деталей в тенях и средних тонах. Метод прост в реализации и показывает хорошие результаты для большинства изображений.

1.9.7 Гиперболическое преобразование

Гиперболическое преобразование усиливает контраст в темных областях изображения. Параметр α позволяет гибко настраивать результат, что делает метод универсальным для разных задач.

2 Task. Профили

2.1 Функция plot_brightness_profile. Построение профиля яркости

Функция строит график изменения яркости вдоль заданной линии на изображении.

2.1.1 Код

```
def plot_brightness_profile(image, line_start, line_end):
    height, width = image.shape[:2]

    num_points = 1000
    x = np.linspace(line_start[0], line_end[0], num_points)
    y = np.linspace(line_start[1], line_end[1], num_points)

    profile = [image[int(y[i])], int(x[i])] for i in range(num_points)]

    plt.figure(figsize=(8, 4))
    plt.plot(profile, color='black')
    plt.title("Профиль яркости")
    plt.xlabel("Позиция вдоль линии")
    plt.ylabel("Яркость")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

2.1.2 Объяснение

Функция принимает изображение и координаты начальной (`line_start`) и конечной (`line_end`) точек линии. Для построения профиля яркости:

- Создается массив точек вдоль линии с использованием `np.linspace`.
- Для каждой точки извлекается значение яркости из изображения.
- Строится график изменения яркости вдоль линии.

2.1.3 Итог

На вход подается изображение и координаты линии. На выходе получается график изменения яркости вдоль этой линии.

2.1.4 Выход



Рис. 17: Штрих код

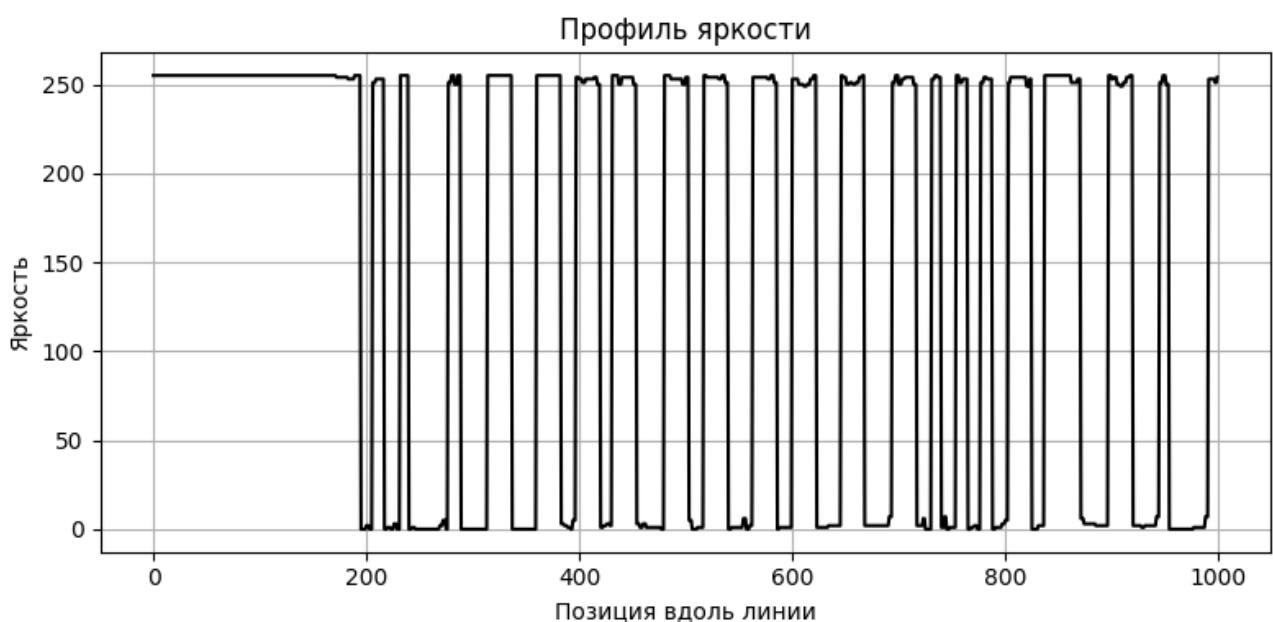


Рис. 18: График профиля яркости

3 Task. Проекции

3.1 Функция render_projections. Построение проекций изображения

Функция строит проекции изображения на горизонтальную (X) и вертикальную (Y) оси, а также визуализирует исходное изображение.

3.1.1 Код

```
def render_projections(img):
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    projection_x = np.sum(gray_img, axis=0) / gray_img.shape[0]
    projection_y = np.sum(gray_img, axis=1) / gray_img.shape[1]

    plt.figure(figsize=(12, 6))

    plt.subplot(2, 2, 1)
    plt.title("Исходное изображение")
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.subplot(2, 2, 2)
    plt.title("Проекция на X")
    plt.plot(projection_x, color='blue')
    plt.xlabel("Позиция по X")
    plt.ylabel("Интенсивность")
    plt.grid(True)

    plt.subplot(2, 2, 3)
    plt.title("Проекция на Y")
    plt.plot(projection_y, range(gray_img.shape[0]), color='red')
    plt.xlabel("Интенсивность")
    plt.ylabel("Позиция по Y")
    plt.grid(True)

    plt.tight_layout()
    plt.show()
```

3.1.2 Объяснение

Функция принимает цветное изображение и выполняет следующие шаги:

- Преобразует изображение в (оттенки серого) с помощью `cv2.cvtColor`.
- Вычисляет проекцию на горизонтальную ось (X) как сумму интенсивностей по строкам, нормированную на высоту изображения:

$$\text{projection_x} = \frac{\sum_{\text{строка}} \text{пиксель}}{\text{высота}}.$$

- Вычисляет проекцию на вертикальную ось (Y) как сумму интенсивностей по столбцам, нормированную на ширину изображения:

$$\text{projection_y} = \frac{\sum_{\text{столбец}} \text{пиксель}}{\text{ширина}}.$$

- Визуализирует исходное изображение и графики проекций на X и Y.

3.1.3 Итог

На вход подается цветное изображение. На выходе получается визуализация исходного изображения и графиков проекций на горизонтальную и вертикальную оси.

3.1.4 Выход

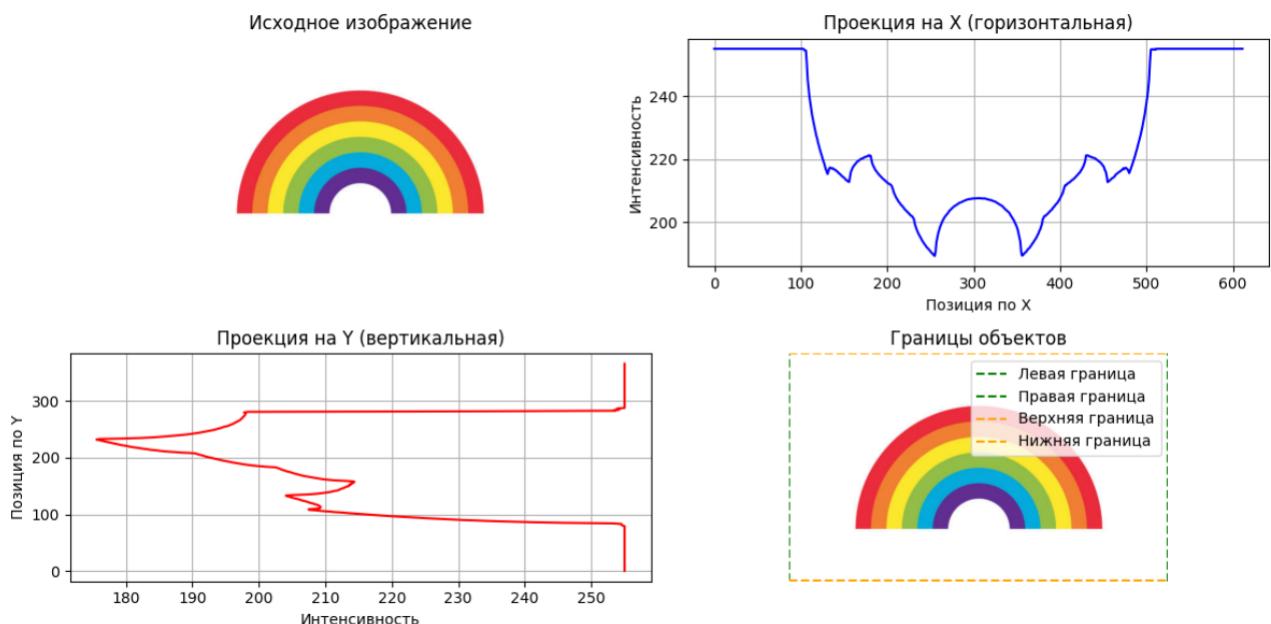


Рис. 19: Исходное изображение и проекции на оси X и Y

4 Вывод

В ходе выполнения лабораторной работы я освоила различные методы преобразования изображений для улучшения их визуального качества и упрощения последующего анализа. Я также изучила работу с профилями и проекциями изображений, что позволило мне эффективно выделять границы и объекты на изображениях.

Для реализации задач я использовала библиотеки OpenCV и Matplotlib, которые значительно упростили процесс обработки и визуализации изображений. Эти инструменты помогли мне лучше понять принципы работы различных преобразований и их влияние на изображения.

Скоро все результаты, включая код для генерации гистограмм и других данных, будут доступны в моем репозитории на GitHub <https://github.com/decadeos/texViz>.