МИНИСТЕРСВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автомное учреждение выщего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»



Институт Прикладной Математики и компютерных наук

02.03.02 «Фундаментальная информатика и информационные технологии»

отчет

Лабораторная работа №2

По дисциплине:

Распознавание образов и компьютерное зрение

**Исполнители:**

Студенты группы 932101 Насутион Рафли Аулиа Ризки

**Руководитель:**

Преподаватель Бакланова Ольга Евгеньевна

Томск - 2024

1. Цель работы

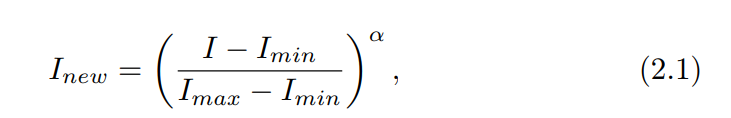
Изучение ключевых особенностей изображений (яркости и формы) и применение этих знаний для их анализа.

2. Гистограммы

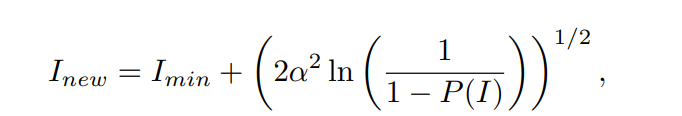
2.1 Задачи

Выбрать произвольное слабоконтрастное изображение. Выполнить выравнивание гистограммы и растяжение контраста, использовать рассмотренные преобразования и встроенные функции пакета MATLAB, с использованием библиотеки OpenCV и языков программирования C++, PYTHON. Сравнить полученные результаты

2.2 Растяжение динамического диапазона и преобразование по закону Рэлея

Если интенсивности пикселей областей интереса находятся в узком динамическом диапазоне, то можно растянуть этот диапазон. Подобные преобразования выполняются согласно следующему выражению:  


где 𝐼 и 𝐼𝑛𝑒𝑤 — массивы значений интенсивностей исходного и нового изображений соответственно; 𝐼𝑚𝑖𝑛 и 𝐼𝑚𝑎𝑥 — минимальное и максимальное значения интенсивностей исходного изображения соответственно; 𝛼 — коэффициент нелинейности.

Преобразование по закону Рэлея осуществляется по следующей формуле:  


где 𝛼 — постоянная, характеризующая гистограмму распределения интенсивностей элементов результирующего изображения.

2.2 Исходное изображение



2.3 Выравнивание гистограммы и растяжение контраста, использовать встроенные функции пакета MATLAB

% Шаг 1: Чтение низкоконтрастного цветного изображения

I = imread('sample\_pict.jpg'); % Загрузка низкоконтрастного цветного изображения

% Шаг 2: Получение размеров изображения и количества каналов

[numRows, numCols, Layers] = size(I);

% Шаг 3: Извлечение отдельных каналов RGB

I\_red = I(:, :, 1); % Извлечение красного канала

I\_green = I(:, :, 2); % Извлечение зеленого канала

I\_blue = I(:, :, 3); % Извлечение синего канала

% Шаг 4: Применение адаптивного выравнивания гистограммы к каждому каналу

I\_red\_eq = adapthisteq(I\_red); % Адаптивное выравнивание гистограммы для красного канала

I\_green\_eq = adapthisteq(I\_green); % Адаптивное выравнивание гистограммы для зеленого канала

I\_blue\_eq = adapthisteq(I\_blue); % Адаптивное выравнивание гистограммы для синего канала

% Шаг 5: Применение растяжения контраста для каждого канала

I\_red\_stretch = imadjust(I\_red\_eq, stretchlim(I\_red\_eq), []); % Автоматическое растяжение контраста для красного канала

I\_green\_stretch = imadjust(I\_green\_eq, stretchlim(I\_green\_eq), []); % Автоматическое растяжение контраста для зеленого канала

I\_blue\_stretch = imadjust(I\_blue\_eq, stretchlim(I\_blue\_eq), []); % Автоматическое растяжение контраста для синего канала

% Шаг 6: Объединение обработанных каналов в цветное изображение

I\_enhanced = cat(3, I\_red\_stretch, I\_green\_stretch, I\_blue\_stretch); % Объединение улучшенных каналов RGB

% Шаг 7: Визуализация исходного изображения, результатов обработки и гистограмм

figure;

% Исходное изображение

subplot(3,4,1), imshow(I), title('Исходное цветное изображение');

% Красный канал и его гистограмма

subplot(3,4,2), imshow(I\_red), title('Красный канал');

subplot(3,4,3), imshow(I\_red\_stretch), title('Красный канал после улучшения');

subplot(3,4,4), imhist(I\_red), title('Гистограмма красного канала');

% Зеленый канал и его гистограмма

subplot(3,4,5), imshow(I\_green), title('Зеленый канал');

subplot(3,4,6), imshow(I\_green\_stretch), title('Зеленый канал после улучшения');

subplot(3,4,7), imhist(I\_green), title('Гистограмма зеленого канала');

% Синий канал и его гистограмма

subplot(3,4,8), imshow(I\_blue), title('Синий канал');

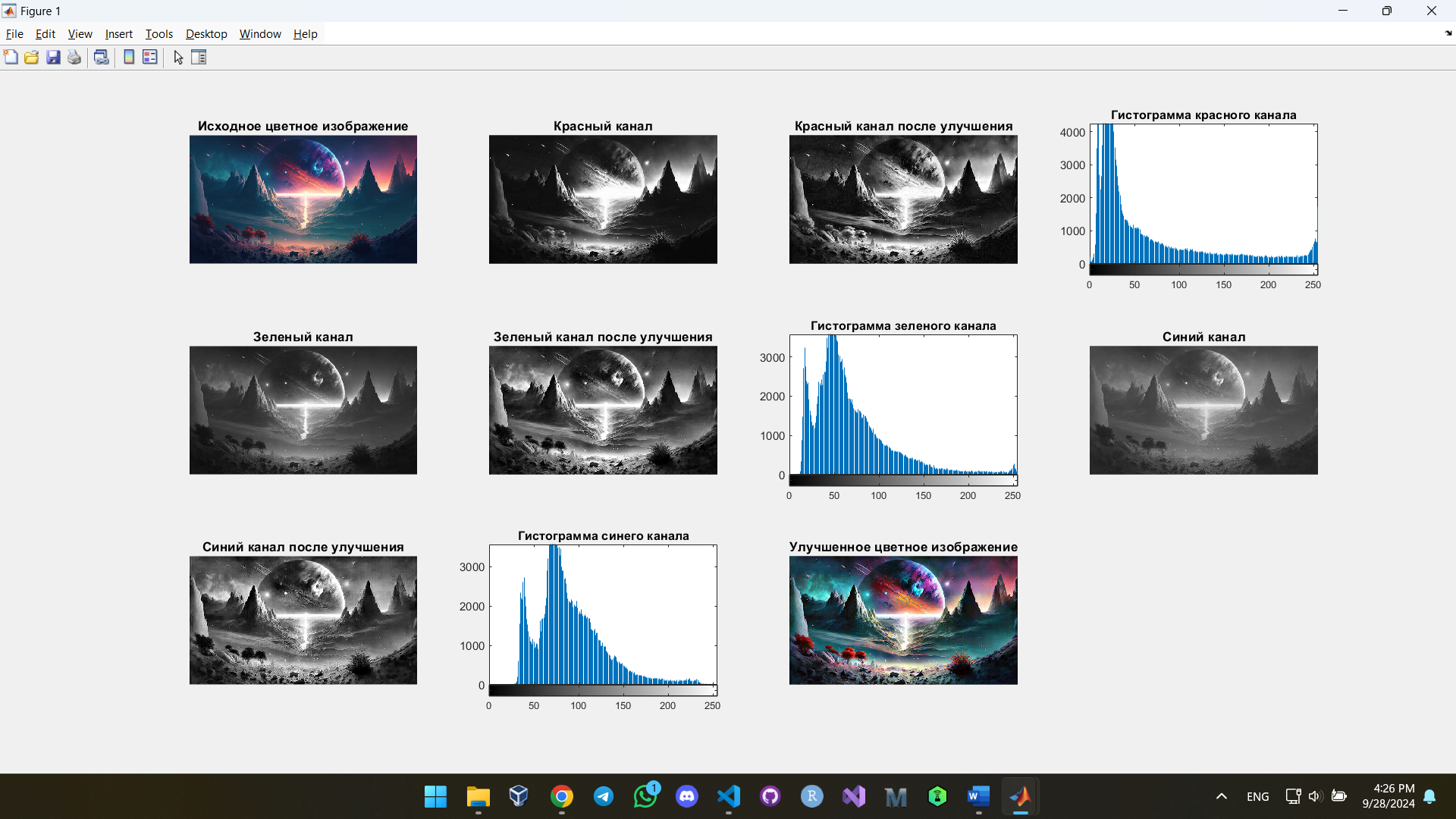
subplot(3,4,9), imshow(I\_blue\_stretch), title('Синий канал после улучшения');

subplot(3,4,10), imhist(I\_blue), title('Гистограмма синего канала');

% Улучшенное изображение

subplot(3,4,11), imshow(I\_enhanced), title('Улучшенное цветное изображение');

**Результат изображения:**



2.4 Выравнивание гистограммы и растяжение контраста, использовать преобразование по закону Рэлея в MATLAB

% Чтение низкоконтрастного цветного изображения

I = imread('Histograms/sample\_pict.jpg');

% Получение размеров изображения

[numRows, numCols, Layers] = size(I);

% Извлечение отдельных каналов RGB

I\_red = I(:, :, 1);

I\_green = I(:, :, 2);

I\_blue = I(:, :, 3);

% Применение адаптивного выравнивания гистограммы к каждому каналу

I\_red\_eq = histogram\_equalization(I\_red, numRows, numCols);

I\_green\_eq = histogram\_equalization(I\_green, numRows, numCols);

I\_blue\_eq = histogram\_equalization(I\_blue, numRows, numCols);

% Применение Рэлеевского преобразования для каждого канала с параметром sigma

sigma = 50; % Параметр σ для Рэлеевского распределения (можно изменять в зависимости от изображения)

I\_red\_stretch = rayleigh\_transform(I\_red\_eq, sigma);

I\_green\_stretch = rayleigh\_transform(I\_green\_eq, sigma);

I\_blue\_stretch = rayleigh\_transform(I\_blue\_eq, sigma);

% Объединение обработанных каналов в цветное изображение

I\_enhanced = cat(3, I\_red\_stretch, I\_green\_stretch, I\_blue\_stretch);

% Визуализация исходного изображения, результатов обработки и гистограмм

figure;

% Исходное изображение

subplot(3, 4, 1), imshow(I), title('Исходное цветное изображение');

% Красный канал и его гистограмма

subplot(3, 4, 2), imshow(I\_red), title('Красный канал');

subplot(3, 4, 3), imshow(I\_red\_stretch), title('Красный канал после Рэлеевского преобразования');

subplot(3, 4, 4), imhist(I\_red), title('Гистограмма красного канала');

% Зеленый канал и его гистограмма

subplot(3, 4, 5), imshow(I\_green), title('Зеленый канал');

subplot(3, 4, 6), imshow(I\_green\_stretch), title('Зеленый канал после Рэлеевского преобразования');

subplot(3, 4, 7), imhist(I\_green), title('Гистограмма зеленого канала');

% Синий канал и его гистограмма

subplot(3, 4, 8), imshow(I\_blue), title('Синий канал');

subplot(3, 4, 9), imshow(I\_blue\_stretch), title('Синий канал после Рэлеевского преобразования');

subplot(3, 4, 10), imhist(I\_blue), title('Гистограмма синего канала');

% Улучшенное изображение

subplot(3, 4, 11), imshow(I\_enhanced), title('Улучшенное цветное изображение');

% Ручная реализация функции выравнивания гистограммы

% Метод: Выравнивание гистограммы (Histogram Equalization)

function I\_eq = histogram\_equalization(I\_channel, numRows, numCols)

% Вычисление гистограммы

hist\_counts = zeros(256, 1);

for i = 1:numRows

for j = 1:numCols

pixel\_value = I\_channel(i, j);

hist\_counts(pixel\_value + 1) = hist\_counts(pixel\_value + 1) + 1;

end

end

% Вычисление кумулятивной функции распределения (CDF)

cdf = cumsum(hist\_counts) / (numRows \* numCols);

cdf\_min = min(cdf(cdf > 0)); % Нахождение минимального ненулевого значения CDF

% Использование CDF для корректировки значений пикселей

I\_eq = zeros(numRows, numCols);

for i = 1:numRows

for j = 1:numCols

pixel\_value = I\_channel(i, j);

I\_eq(i, j) = round((cdf(pixel\_value + 1) - cdf\_min) / (1 - cdf\_min) \* 255);

end

end

I\_eq = uint8(I\_eq); % Преобразование обратно в тип uint8

end

% Ручная реализация Рэлеевского преобразования (распределение Рэлея)

function I\_out = rayleigh\_transform(I\_channel, sigma)

% Преобразование по закону Рэлея

I\_channel = double(I\_channel); % Преобразование в тип double для расчётов

I\_out = (I\_channel / sigma^2) .\* exp(-I\_channel.^2 / (2 \* sigma^2));

% Нормализация к диапазону [0, 255]

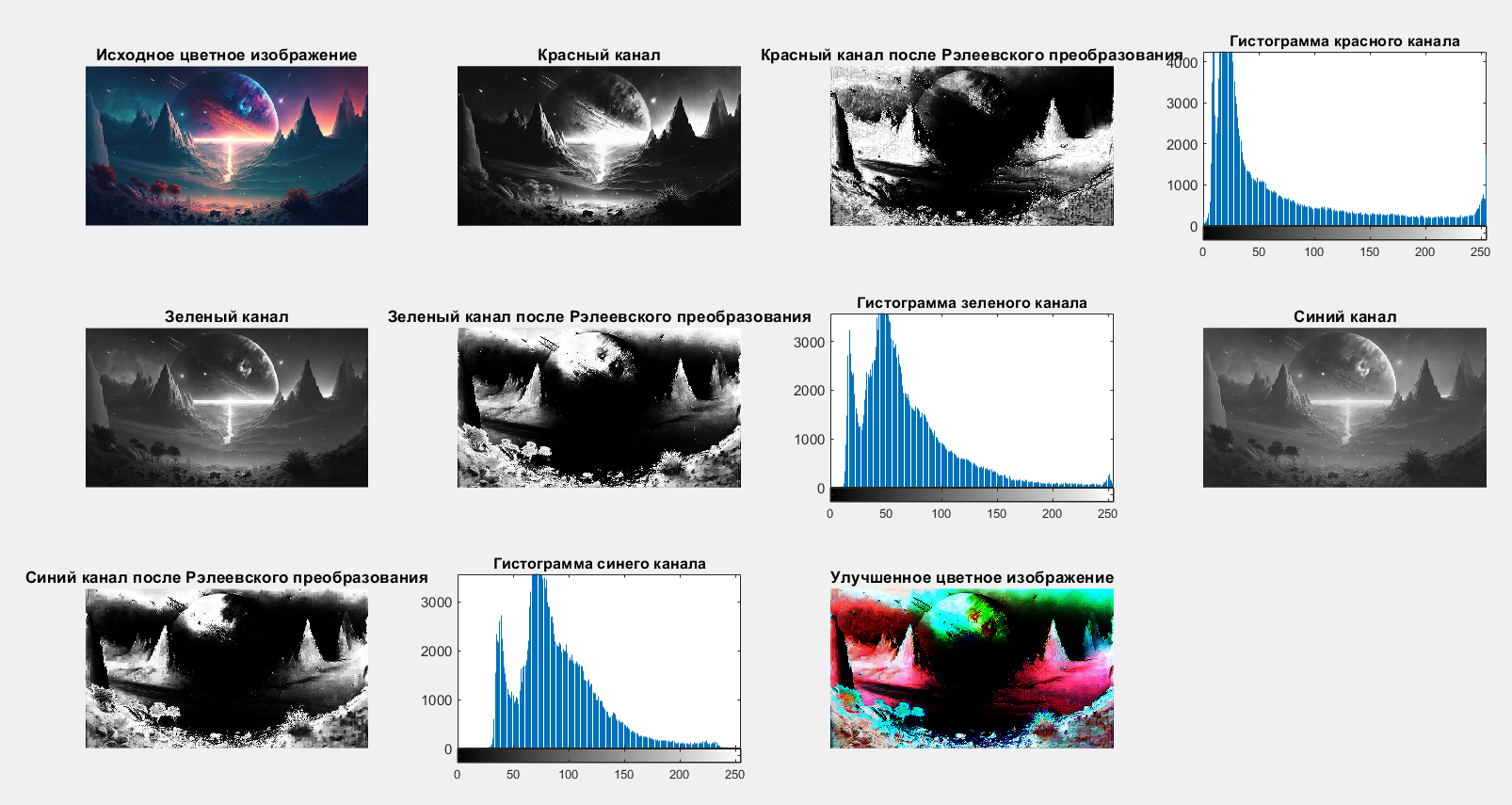
I\_out = I\_out - min(I\_out(:)); % Вычитаем минимальное значение

I\_out = I\_out / max(I\_out(:)) \* 255; % Масштабируем к диапазону [0, 255]

I\_out = uint8(I\_out); % Преобразуем обратно в тип uint8

end

**Результат изображения:**



2.5 С использованием библиотеки OpenCV и языков программирования Python

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Функция для разделения и возврата каналов RGB

def split\_channels(image):

    return image[:, :, 2], image[:, :, 1], image[:, :, 0]

# Чтение низкоконтрастного цветного изображения

image = cv2.imread('sample\_pict.jpg')

# Получение размеров изображения

rows, cols, channels = image.shape

# Извлечение каналов RGB с помощью функции

red\_channel, green\_channel, blue\_channel = split\_channels(image)

# Переписанная функция выравнивания гистограммы

def equalize\_histogram(channel):

    histogram, \_ = np.histogram(channel.flatten(), bins=256, range=[0, 256])

    cdf = histogram.cumsum() / histogram.sum()

    cdf\_min = cdf[np.nonzero(cdf)].min()  # Использование nonzero вместо прямого условия

    equalized\_channel = np.interp(channel, np.arange(256), (cdf - cdf\_min) / (1 - cdf\_min) \* 255).astype(np.uint8)

    return equalized\_channel

# Применение выравнивания гистограммы к каждому каналу

red\_eq = equalize\_histogram(red\_channel)

green\_eq = equalize\_histogram(green\_channel)

blue\_eq = equalize\_histogram(blue\_channel)

# Изменённая функция Рэлеевского преобразования для упрощения

def apply\_rayleigh\_transform(channel):

    scale\_factor = 255 / np.log1p(255)  # Использование np.log1p для улучшенной стабильности

    transformed\_channel = scale\_factor \* np.log1p(channel.astype(np.float32))

    return np.clip(transformed\_channel, 0, 255).astype(np.uint8)

# Применение Рэлеевского преобразования к каждому каналу

red\_transformed = apply\_rayleigh\_transform(red\_eq)

green\_transformed = apply\_rayleigh\_transform(green\_eq)

blue\_transformed = apply\_rayleigh\_transform(blue\_eq)

# Объединение каналов обратно в улучшенное изображение

enhanced\_image = cv2.merge([blue\_transformed, green\_transformed, red\_transformed])

# Построение графиков в изменённой компоновке

fig, ax = plt.subplots(3, 4, figsize=(14, 8))

# Исходное изображение

ax[0, 0].imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

ax[0, 0].set\_title('Исходное изображение')

# Красный канал и его преобразования

ax[0, 1].imshow(red\_channel, cmap='gray')

ax[0, 1].set\_title('Красный канал')

ax[0, 2].imshow(red\_transformed, cmap='gray')

ax[0, 2].set\_title('Красный канал после Рэлеевского преобразования')

ax[0, 3].hist(red\_channel.ravel(), bins=256, range=[0, 256], color='r')

ax[0, 3].set\_title('Гистограмма красного канала')

# Зелёный канал и его преобразования

ax[1, 1].imshow(green\_channel, cmap='gray')

ax[1, 1].set\_title('Зелёный канал')

ax[1, 2].imshow(green\_transformed, cmap='gray')

ax[1, 2].set\_title('Зелёный канал после Рэлеевского преобразования')

ax[1, 3].hist(green\_channel.ravel(), bins=256, range=[0, 256], color='g')

ax[1, 3].set\_title('Гистограмма зелёного канала')

# Синий канал и его преобразования

ax[2, 1].imshow(blue\_channel, cmap='gray')

ax[2, 1].set\_title('Синий канал')

ax[2, 2].imshow(blue\_transformed, cmap='gray')

ax[2, 2].set\_title('Синий канал после Рэлеевского преобразования')

ax[2, 3].hist(blue\_channel.ravel(), bins=256, range=[0, 256], color='b')

ax[2, 3].set\_title('Гистограмма синего канала')

# Улучшенное изображение

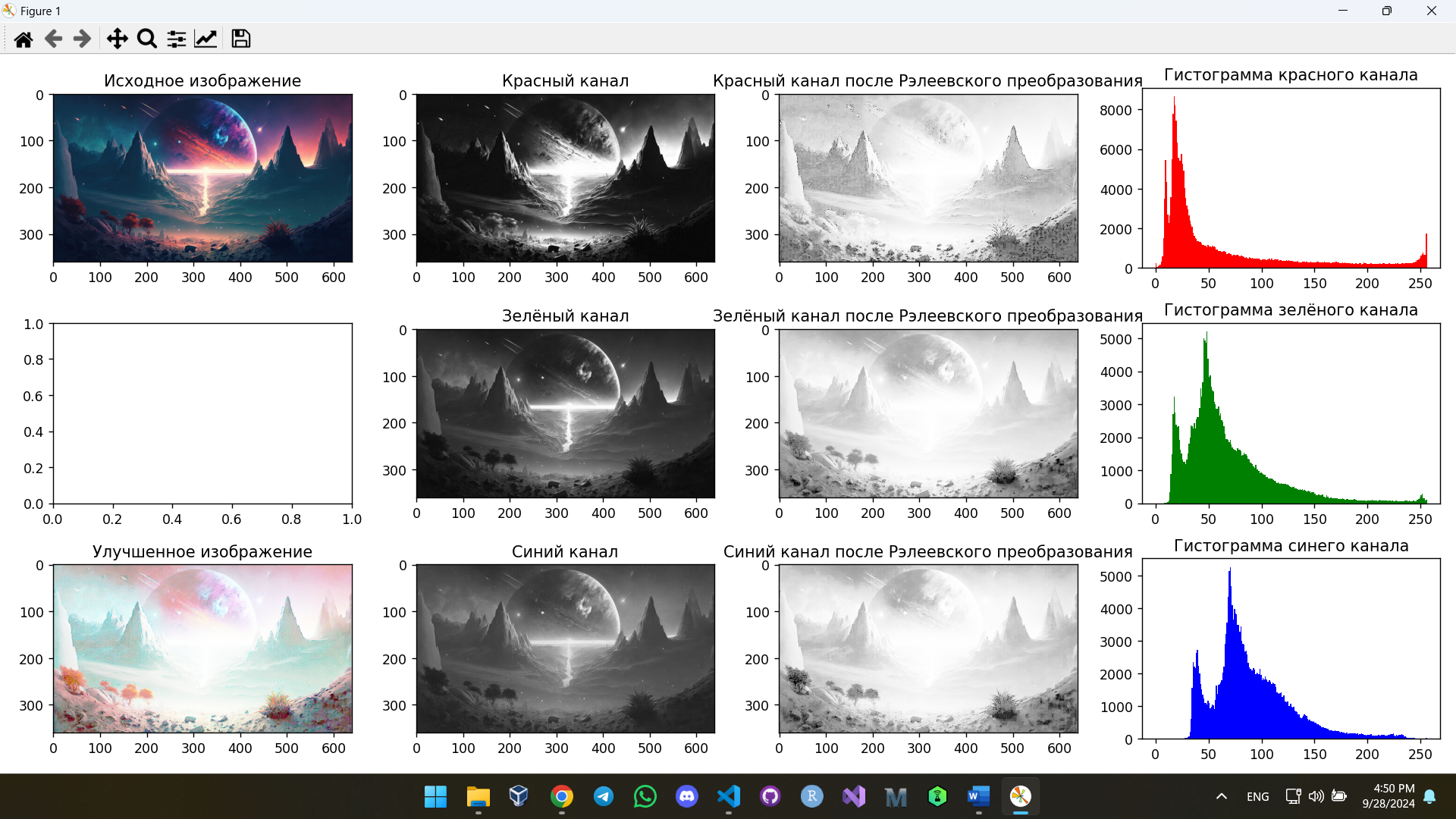
ax[2, 0].imshow(cv2.cvtColor(enhanced\_image, cv2.COLOR\_BGR2RGB))

ax[2, 0].set\_title('Улучшенное изображение')

plt.tight\_layout()

plt.show()

**Результат изображения:**



2.6. С использованием библиотеки OpenCV и языков программирования C++

#include <opencv2/opencv.hpp>

#include <iostream>

#include <vector>

#include <cmath> // для функции log

#include <numeric> // для функции accumulate

using namespace cv;

using namespace std;

// Функция гистограммного выравнивания

Mat histogramEqualization(const Mat& I\_channel) {

// Вычисляем гистограмму

int histSize = 256; // Количество уровней яркости

float range[] = { 0, 256 }; // Диапазон значений

const float\* histRange = { range };

Mat hist;

calcHist(&I\_channel, 1, 0, Mat(), hist, 1, &histSize, &histRange, true, false); // Вычисление гистограммы

// Вычисляем функцию накопления (CDF)

vector<float> cdf(histSize, 0);

cdf[0] = hist.at<float>(0);

for (int i = 1; i < histSize; i++) {

cdf[i] = cdf[i - 1] + hist.at<float>(i); // Накопление значений гистограммы

}

// Нормализуем функцию накопления

float cdf\_min = \*min\_element(cdf.begin(), cdf.end()); // Находим минимальное значение CDF

Mat I\_eq = I\_channel.clone(); // Копируем исходный канал

for (int i = 0; i < I\_channel.rows; i++) {

for (int j = 0; j < I\_channel.cols; j++) {

if (cdf\_min < 1) { // Избегаем деления на ноль

I\_eq.at<uchar>(i, j) = saturate\_cast<uchar>((cdf[I\_channel.at<uchar>(i, j)] - cdf\_min) / (1 - cdf\_min) \* 255);

}

else {

I\_eq.at<uchar>(i, j) = I\_channel.at<uchar>(i, j); // Без изменений, если cdf\_min слишком высоко

}

}

}

return I\_eq; // Возвращаем выровненный канал

}

// Функция преобразования Релея

Mat rayleighTransform(const Mat& I\_eq) {

Mat result = I\_eq.clone(); // Копируем выровненный канал

double c = 255.0 / log(1.0 + 255.0); // Вычисляем коэффициент c

for (int i = 0; i < I\_eq.rows; i++) {

for (int j = 0; j < I\_eq.cols; j++) {

result.at<uchar>(i, j) = saturate\_cast<uchar>(c \* log(1 + I\_eq.at<uchar>(i, j)));

}

}

return result; // Возвращаем результат преобразования Релея

}

// Функция для отображения гистограммы

void showHistogram(const Mat& channel, const string& windowName) {

int histSize = 256; // Количество уровней яркости

float range[] = { 0, 256 }; // Диапазон значений

const float\* histRange = { range };

Mat hist;

// Вычисляем гистограмму

calcHist(&channel, 1, 0, Mat(), hist, 1, &histSize, &histRange, true, false);

// Нормализуем гистограмму

Mat histImage = Mat::zeros(200, 256, CV\_8UC1);

normalize(hist, hist, 0, histImage.rows, NORM\_MINMAX, -1, Mat());

// Рисуем гистограмму

for (int i = 1; i < histSize; i++) {

line(histImage, Point(i - 1, 200 - cvRound(hist.at<float>(i - 1))),

Point(i, 200 - cvRound(hist.at<float>(i))), Scalar(255), 2, 8, 0);

}

imshow(windowName, histImage); // Отображаем гистограмму

}

int main() {

// Mat I = imread("E:\\Desktop\\work-space\\CV\\lab2\\Histograms\\lab2\_C++\\123.jpg"); // Загружаем изображение

Mat I = imread("D:\\ИПМКН\\Семестр\_7\\Распознавание Образов\\Lab2\\Histograms\\sample\_pict.jpg"); // Загружаем изображение

if (I.empty()) {

cout << "Ошибка: Не удалось открыть изображение. Пожалуйста, проверьте путь и файл." << endl;

return -1;

}

// Извлекаем каналы BGR

vector<Mat> channels(3);

split(I, channels); // Разделяем изображение на каналы B, G, R

// Применяем гистограммное выравнивание к каждому каналу

Mat I\_red\_eq = histogramEqualization(channels[2]); // Красный

Mat I\_green\_eq = histogramEqualization(channels[1]); // Зеленый

Mat I\_blue\_eq = histogramEqualization(channels[0]); // Синий

// Применяем преобразование Релея к каждому каналу

Mat I\_red\_stretch = rayleighTransform(I\_red\_eq);

Mat I\_green\_stretch = rayleighTransform(I\_green\_eq);

Mat I\_blue\_stretch = rayleighTransform(I\_blue\_eq);

// Объединяем обработанные каналы в цветное изображение

Mat I\_enhanced;

merge(vector<Mat>{I\_blue\_stretch, I\_green\_stretch, I\_red\_stretch}, I\_enhanced);

// Отображаем оригинальное и улучшенное изображения

imshow("Оригинальное изображение", I);

imshow("Улучшенное изображение", I\_enhanced);

// Отображаем оригинальные и улучшенные каналы

imshow("Оригинальный красный канал", channels[2]);

imshow("Улучшенный красный канал", I\_red\_stretch);

imshow("Оригинальный зеленый канал", channels[1]);

imshow("Улучшенный зеленый канал", I\_green\_stretch);

imshow("Оригинальный синий канал", channels[0]);

imshow("Улучшенный синий канал", I\_blue\_stretch);

// Отображаем гистограммы для каждого канала

showHistogram(channels[2], "Гистограмма красного канала");

showHistogram(channels[1], "Гистограмма зеленого канала");

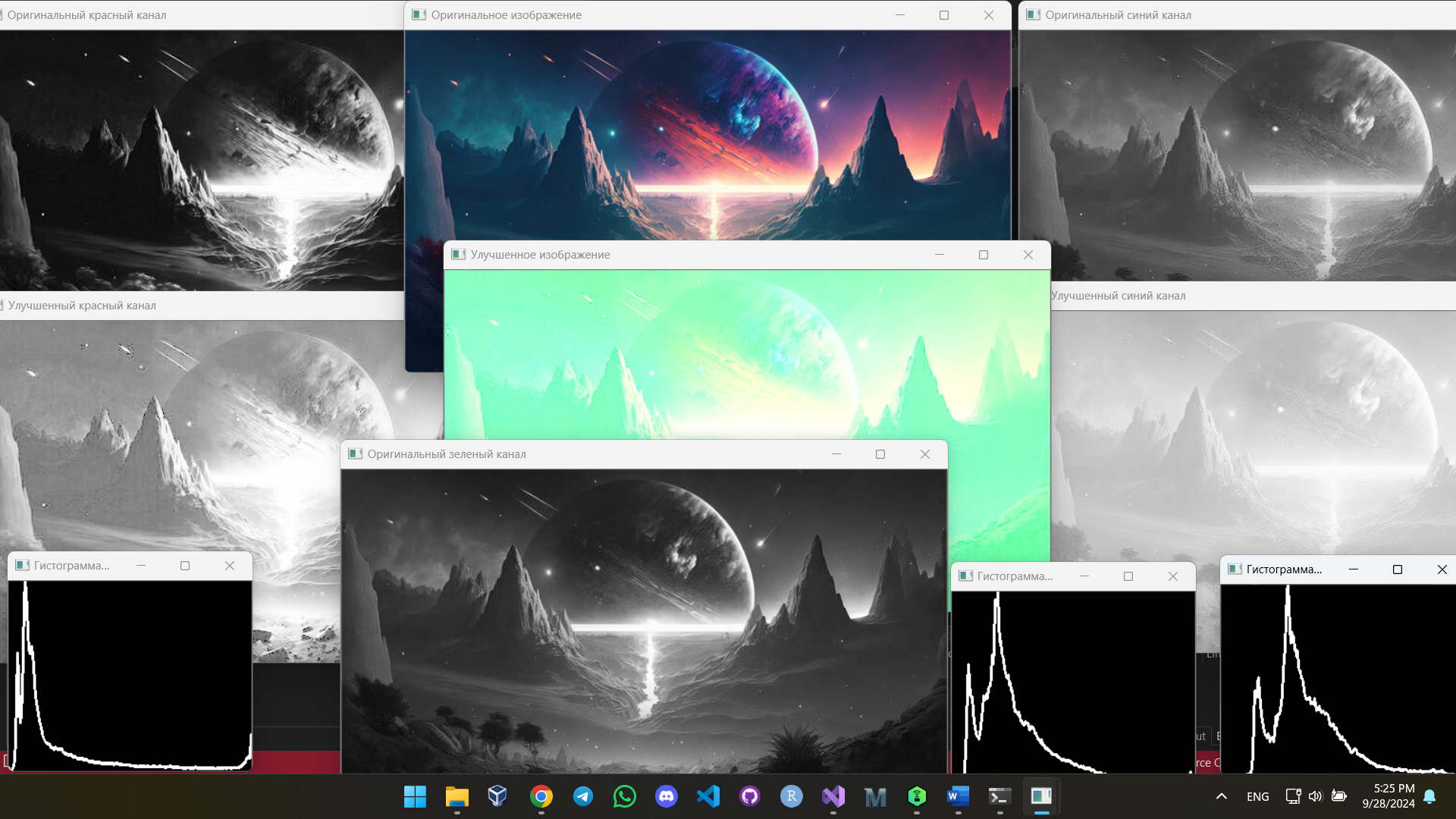
showHistogram(channels[0], "Гистограмма синего канала");

waitKey(0); // Ожидание нажатия клавиши

return 0;

}

**Результат изображения:**



3. Проекции

3.1 Задачи

Выбрать произвольное изображение, содержащее монотонные области и выделяющиеся объекты. Произвести построение проекций изображения на вертикальную и горизонтальную оси. Определить границы областей объектов.

3.2 Исходное изображение



3.3 Реализация в Matlab

% 1. Загрузка изображения

I = imread('wall.jpg');

% 2. Получение размеров изображения

[numRows, numCols, numChannels] = size(I);

% 3. Инициализация результатов горизонтальной и вертикальной проекций

Proj\_H = zeros(numRows, 1); % Для хранения горизонтальной проекции (сумма значений пикселей в каждой строке)

Proj\_V = zeros(numCols, 1); % Для хранения вертикальной проекции (сумма значений пикселей в каждом столбце)

% 4. Расчет горизонтальной проекции (построчное суммирование значений пикселей в каналах RGB)

for i = 1:numRows

Proj\_H(i) = sum(I(i, :, 1)) + sum(I(i, :, 2)) + sum(I(i, :, 3)); % Суммирование значений в каналах RGB

end

% 5. Расчет вертикальной проекции (постолбцовое суммирование значений пикселей в каналах RGB)

for j = 1:numCols

Proj\_V(j) = sum(I(:, j, 1)) + sum(I(:, j, 2)) + sum(I(:, j, 3)); % Суммирование значений в каналах RGB

end

% 6. Построение графиков горизонтальной и вертикальной проекций

figure;

subplot(2, 1, 1);

plot(Proj\_H); % Построение графика горизонтальной проекции

title('Горизонтальная проекция');

xlabel('Номер строки');

ylabel('Сумма интенсивностей пикселей');

subplot(2, 1, 2);

plot(Proj\_V); % Построение графика вертикальной проекции

title('Вертикальная проекция');

xlabel('Номер столбца');

ylabel('Сумма интенсивностей пикселей');

% 7. Определение границ объекта (на основе простого метода порогового значения)

threshold\_H = max(Proj\_H) \* 0.5; % Установка порога для горизонтальной проекции на уровне 50% от максимального значения

object\_rows = find(Proj\_H > threshold\_H); % Нахождение строк, где проекция превышает порог

threshold\_V = max(Proj\_V) \* 0.5; % Установка порога для вертикальной проекции на уровне 50% от максимального значения

object\_cols = find(Proj\_V > threshold\_V); % Нахождение столбцов, где проекция превышает порог

% 8. Отметка границ объекта на изображении

figure;

imshow(I); % Отображение исходного изображения

hold on;

rectangle('Position', [min(object\_cols), min(object\_rows), ...

max(object\_cols) - min(object\_cols), ...

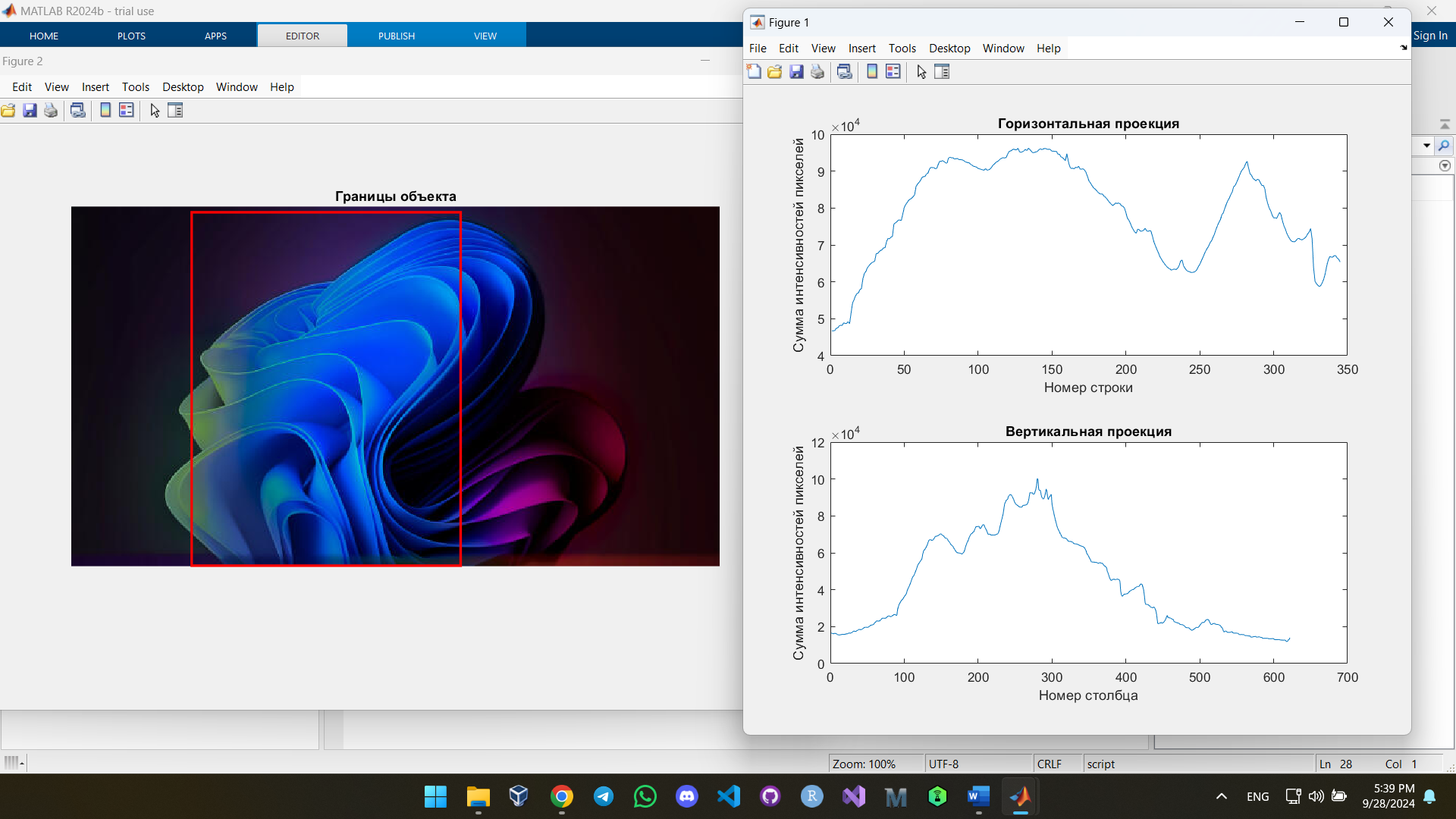
max(object\_rows) - min(object\_rows)], ...

'EdgeColor', 'r', 'LineWidth', 2); % Рисование красной рамки для выделения границ объекта

hold off;

title('Границы объекта');

**Результат изображения:**



4. Профили

4.1 Задачи

Выбрать произвольное изображение, содержащие штрих-код. Выполнить построение профиля изображения вдоль штрих-кода.

4.2 Исходное изображение



4.3 Реализация в Matlab

% 1. Чтение изображения

img = imread('barcode\_pict.jpg');

% 2. Преобразование изображения в градации серого

grayImg = im2gray(img);

% 3. Выбор строки, где находится штрих-код

% Предполагаем, что мы будем рисовать контур по средней строке

row = round(size(grayImg, 1) / 2);

profile = grayImg(row, :);

% 4. Построение профиля

figure;

plot(profile);

title('Профиль изображения вдоль штрих-кода');

xlabel('Позиция пикселя');

ylabel('Интенсивность');

**Результат изображения:**



5. Вывод

В результате этой работе удалось разработать эффективные алгоритмы обработки изображений, позволяющие улучшить качество изображений с низким контрастом и выделить интересующие объекты. Было проведено сравнительное исследование различных методов обработки на языках MATLAB, C++ и Python. Полученные результаты демонстрируют высокую эффективность разработанных алгоритмов.