Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ НА PYTHON

Отчет по лабораторной работе по дисциплине

«Разработка веб-сервисов для научных и прикладных задач»

Вариант №7

Студент гр. \_\_\_\_\_

Е.С. Селезнев

(И. О. Фамилия)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

Руководитель

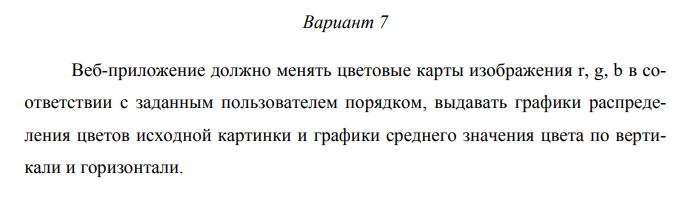
доцент каф. АСУ,

канд. техн. наук

А. Я. Суханов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

Томск 2025



Оглавление

Элементы оглавления не найдены.

* 1. Структура проекта

На Рис. 1 и Рис. 2 показан каталог и содержимое проекта, директория templates используется для хранения шаблонов html, в данном проекте мною было принято решение реализовать каркас сайта в отельном файле, не используя декоратор для его реализации. Static и дочерний каталог images используются для хранения медиа контента, в представленном проекте — jpg картинок.

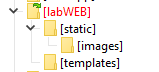


Рис. 1. Директория проекта

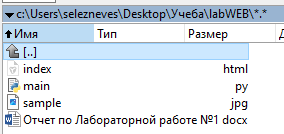


Рис. 2. Файлы проекта

* 1. Разбор элементов интерфейса

На Рис. 3 изображен веб-интерфейс получившегося проекта. Для наглядности работы веб-приложения я использовал изображение с множеством цветов в блоке «Результат».

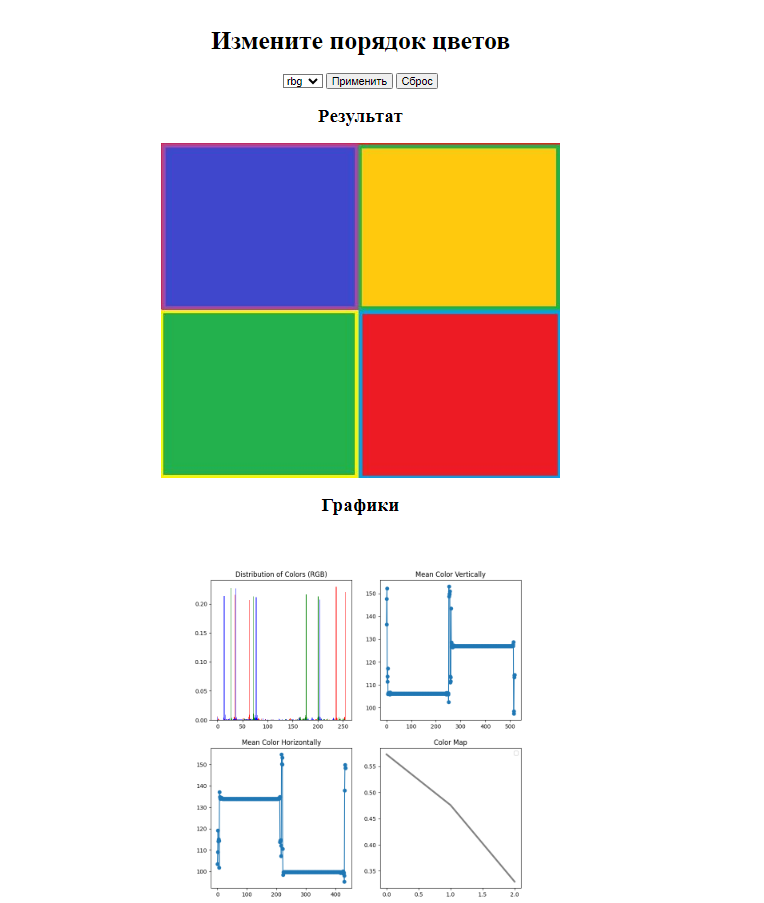


Рис. 3.

Рассмотрим подробнее элементы веб-приложения, на Рис. 4 мы можем увидеть элементы управления цветовой раскладкой изображения, выбрав в списке цветовое смещение и нажав кнопку «применить» происходит преобразование цветового спектра исходного изображения, которое представлено на Рис. 5. Кнопка «Сброс» возвращает изменённое изображение к исходному виду.

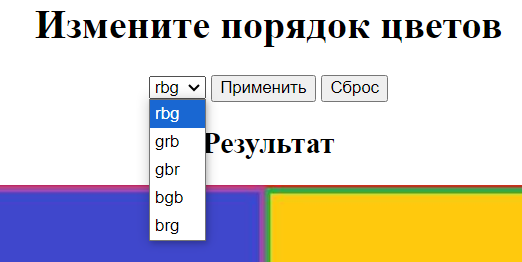


Рис. 4. Список цветового смещения

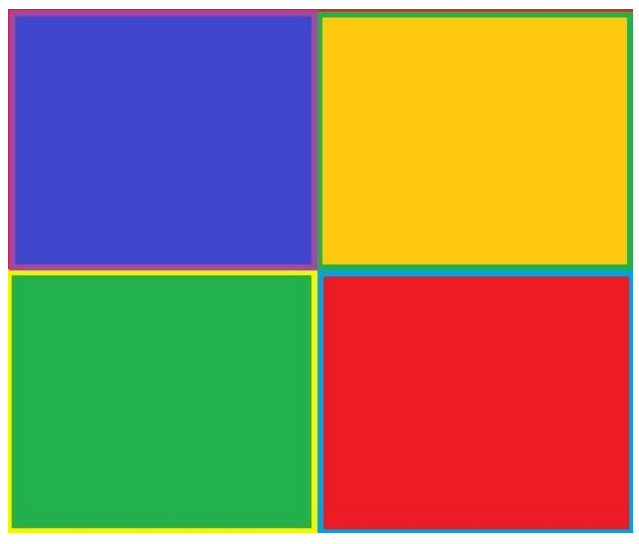


Рис. 5. Исходное изображение

Элементы блока графиков изображены на Рис.6.

* График №1 «Distribution of Colors (RGB)» — это гистограмма, которая показывает распределение цветов в изображении по трем каналам: Red (Красный), Green (Зелёный) и Blue (Синий). Эта гистограмма отображает частоту встречаемости каждого уровня интенсивности (от 0 до 255) для каждого цветового канала. Вертикальная ось показывает количество пикселей с определенным уровнем интенсивности в каждом канале. Чем выше столбец на гистограмме, тем больше пикселей имеет соответствующий уровень интенсивности. Горизонтальная ось отображает уровни интенсивности для каждого цветового канала (Red, Green, Blue) от 0 до 255.
* График №2 «Mean Color Horizontally» и график №3 «Mean Color Vertically» — это графики, которые показывают средние значения цветов по горизонтали и вертикали изображения соответственно. Они помогают понять, как меняется средний цвет вдоль строк и столбцов изображения. **Horizontal Mean Color Graph** (график среднего цвета по горизонтали) отображает средние значения цветов по каждому столбцу изображения. Вертикальная ось показывает значения среднего цвета (от 0 до 255), а горизонтальная ось соответствует колонкам изображения. Если график среднего цвета по горизонтали показывает плавное увеличение значений от левого края к правому, это может означать постепенное осветление изображения слева направо. **Vertical Mean Color Graph** (график среднего цвета по вертикали) отображает средние значения цветов по каждой строке изображения. Вертикальная ось снова показывает значения среднего цвета (от 0 до 255), а горизонтальная ось соответствует строкам изображения. Если график среднего цвета по вертикали показывает снижение значений снизу-вверх, это может указывать на затемнение изображения от нижней части к верхней.
* График №4 «Color map» — это графическое представление, которое показывает, как различные уровни яркости или насыщенности цвета распределены по всему изображению. Цветовая карта отображает соответствие между значениями пикселей и цветами, помогая визуализировать, какие участки изображения имеют высокие или низкие значения яркости. Если большая часть изображения отображается темно-синими оттенками на цветовой карте, это может указывать на то, что изображение в целом слишком темное, центральная область изображения отображается ярко-желтой или красной, это может означать, что эта область значительно ярче остальной части изображения.

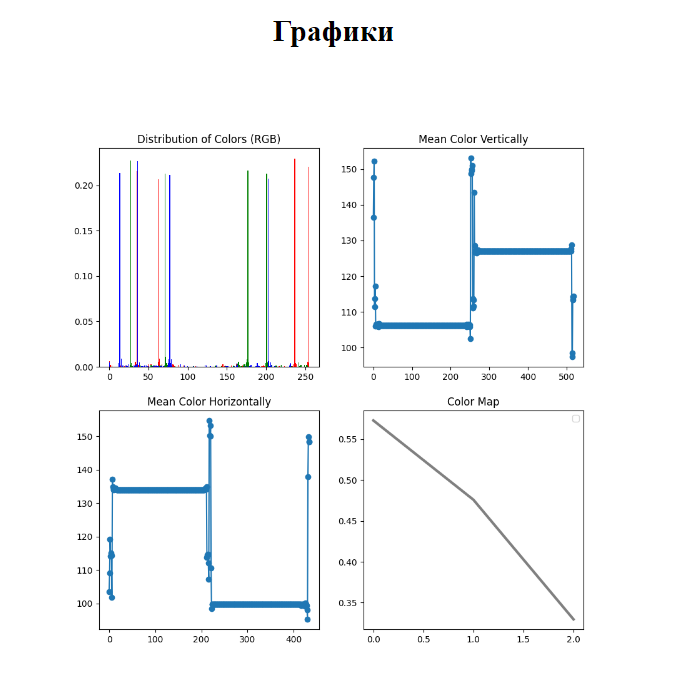


Рис. 6. Графики

* 1. Пример работы

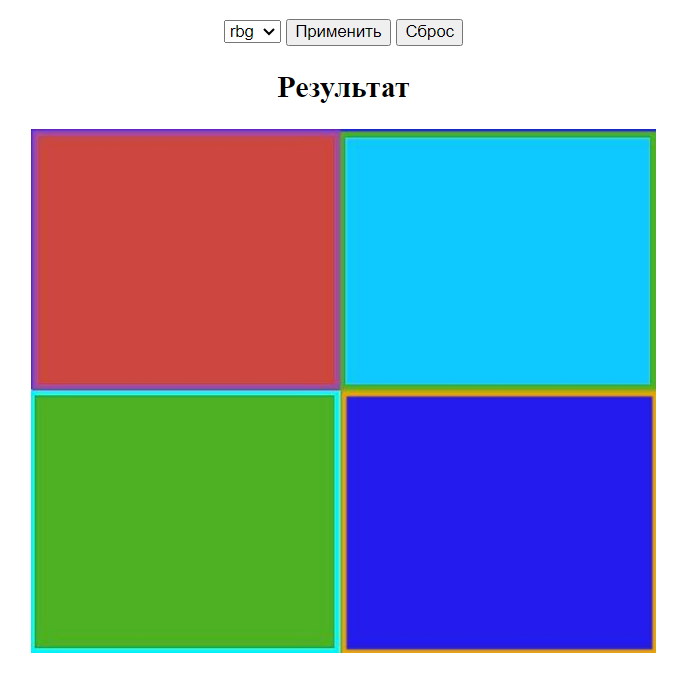


Рис. 7. RBG

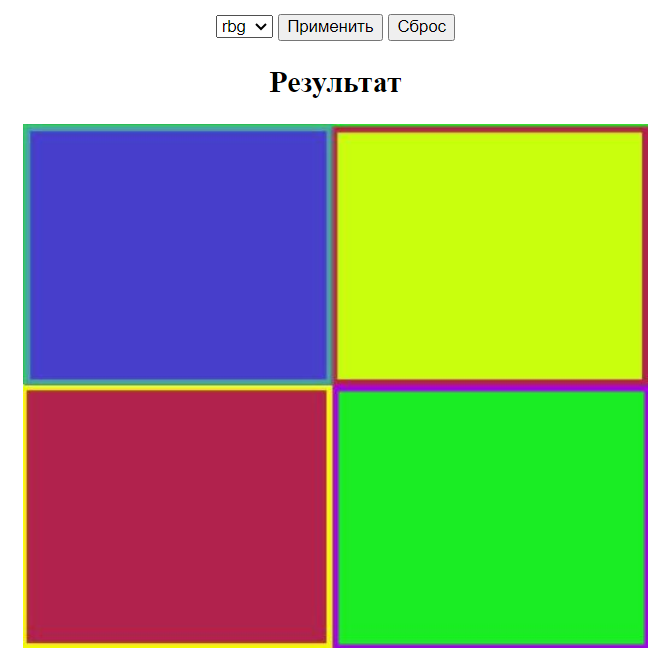


Рис.8 GRB



Рис. 9. GBR

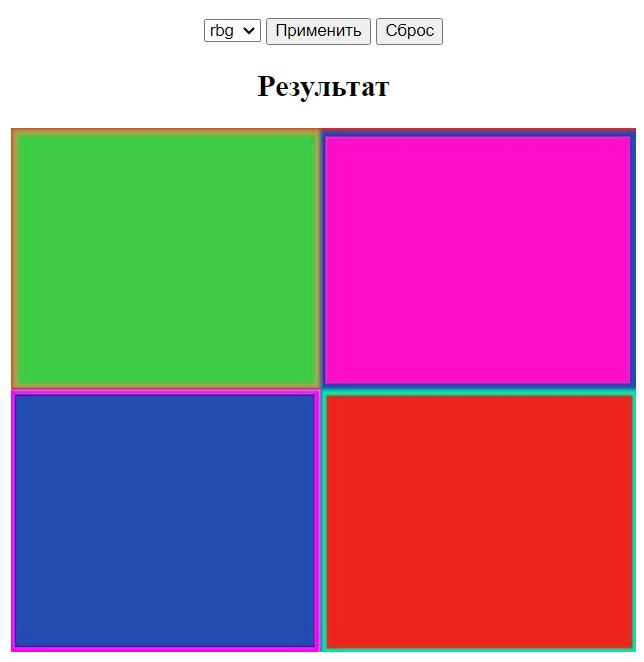


Рис. 10. BGB

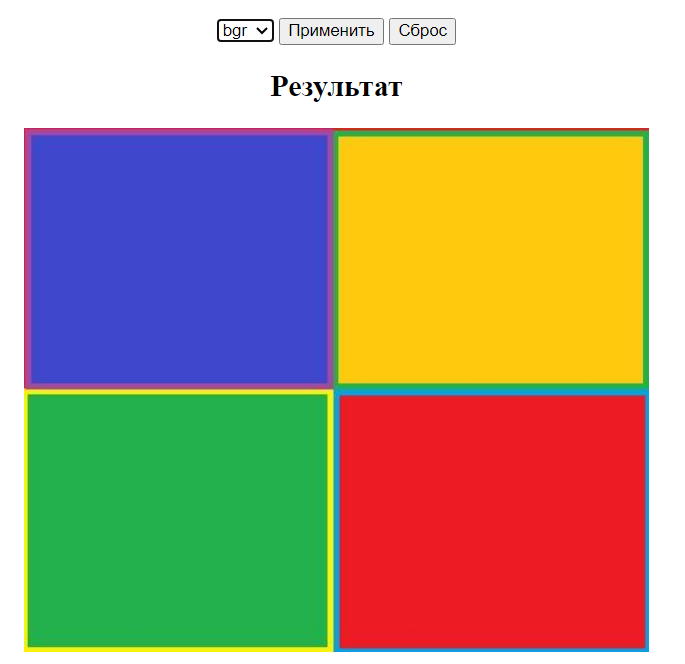


Рис. 11. BGR

На выше представленных рисунках можно увидеть смещение цветовой гаммы, но происходит это на всех канал, кроме BRG, использование BGR вместо RGB не влияет на внешний вид изображения, поскольку оно просто переворачивает порядок цветовых каналов, изменятся только их позиции в цветовой модели изображения.

* 1. Приложение А. Код Python

import io

import base64

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/', methods=['GET', 'POST'])

def index():

    '''

    Функция обработки и представления веб-интерфейса

    '''

    graphs = None  # Инициализация переменной графиков

    encoded\_img = None  # Инициализация переменной изображения

    if request.method == 'POST':

        order = request.form.getlist('order')

        img = Image.open('static/images/sample.jpg') # Использование изображения из директории images

        img\_arr = np.array(img)

        original\_img\_arr = img\_arr.copy() # Создание копии для сброса изображения

        # Изменение порядка цветовых каналов

        if order == ['rbg']:

            img\_arr = img\_arr[..., ::-1]

        elif order == ['grb']:

            img\_arr = img\_arr[..., [1, 0, 2]]

        elif order == ['gbr']:

            img\_arr = img\_arr[..., [1, 2, 0]]

        elif order == ['bgb']:

            img\_arr = img\_arr[..., [0, 2, 1]]

        elif order == ['bgr']:

            img\_arr = img\_arr[..., [0, 1, 2]]

        elif order == ['rgb']:

            img\_arr = original\_img\_arr

        else:

            pass  # Порядок не изменяется

        # Разделение изображения на цветовые каналы

        r, g, b = img\_arr[:, :, 0], img\_arr[:, :, 1], img\_arr[:, :, 2]

        out = io.BytesIO()

        Image.fromarray(img\_arr).save(out, format="JPEG")

        out.seek(0)

        encoded\_img = base64.b64encode(out.read()).decode('utf-8')

        # Генерация графиков

        fig, ax = plt.subplots(2, 2, figsize=(10, 10))

        # Гистограммы для каждого цветового канала

        hist\_r = ax[0][0].hist(r.flatten(), bins=256, range=(0, 255), density=True, color='r')

        hist\_g = ax[0][0].hist(g.flatten(), bins=256, range=(0, 255), density=True, color='g')

        hist\_b = ax[0][0].hist(b.flatten(), bins=256, range=(0, 255), density=True, color='b')

        ax[0][0].set\_title('Distribution of Colors (RGB)')

        means\_vert = [img\_arr[:, i].mean() for i in range(img\_arr.shape[1])]

        means\_horiz = [img\_arr[i, :].mean() for i in range(img\_arr.shape[0])]

        # Преобразование изображения в двумерный массив

        img\_arr\_flat = img\_arr.reshape(-1, img\_arr.shape[-1])

        # Нормализация значений пикселей (0-255 -> 0-1)

        norm\_img\_arr = img\_arr\_flat / 255

        # Создание цветовой карты

        means\_colors = [norm\_img\_arr[:, i].mean() for i in range(norm\_img\_arr.shape[1])]

        ax[0][1].plot(means\_vert, label='Vertical Mean', marker='o')

        ax[0][1].set\_title('Mean Color Vertically')

        ax[1][0].plot(means\_horiz, label='Horizontal Mean', marker='o')

        ax[1][0].set\_title('Mean Color Horizontally')

        ax[1][1].plot(means\_colors, color='gray', linewidth=3)

        ax[1][1].set\_title('Color Map')

        plt.legend()

        graph\_png = io.BytesIO()

        plt.savefig(graph\_png, format='png')

        graph\_png.seek(0)

        encoded\_graph1 = base64.b64encode(graph\_png.read()).decode('utf-8')

        graphs = [encoded\_graph1]  # Обновление переменной перед возвратом функции

    return render\_template('index.html', original\_img=encoded\_img, graphs=graphs)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True) # debug запуск

* 1. Приложение Б. разметка страницы HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Web App</title>

    <style>

        .container {

            display: flex;

            justify-content: center;

            align-items: center;

            flex-direction: column;

            margin-top: 50px;

        }

        img {

            max-width: 500px;

            max-height: 500px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Измените порядок цветов</h1>

        <form method="post">

            <select name="order">

                <option value="rbg">rbg</option>

                <option value="grb">grb</option>

                <option value="gbr">gbr</option>

                <option value="bgb">bgb</option>

                <option value="bgr">bgr</option>

            </select>

            <button type="submit">Применить</button>

            <button type="submit" name="order" value="rgb">Сброс</button>

        </form>

        <h2>Результат</h2>

        <img src="data:image/jpeg;base64,{{ original\_img }}" alt="Result Image">

        <h2>Графики</h2>

        <img src="data:image/png;base64,{{ graphs[0] }}" alt="Graph 1">

    </div>

</body>

</html>