Работа 2. Исследование каналов и JPEG-сжатия

автор: Дворядкин К.А.
дата: 2022-02-23T23:59:35

GitHub

Задание

1. В качестве тестового использовать изображение data/cross_0256x0256.png
2. Сохранить тестовое изображение в формате JPEG с качеством 25%.
3. Используя cv::merge и cv::split сделать "мозаику" с визуализацией каналов для исходного тестового изображения и JPEG-версии тестового изображения

- левый верхний - трехканальное изображение
- левый нижний - монохромная (черно-зеленая) визуализация канала G
- правый верхний - монохромная (черно-красная) визуализация канала R
- правый нижний - монохромная (черно-синяя) визуализация канала B

4. Результы сохранить для вставки в отчет
5. Сделать мозаику из визуализации гистограммы для исходного тестового изображения и JPEG-версии тестового изображения, сохранить для вставки в отчет.

Результаты

Рис. 1. Тестовое изображение после сохранения в формате JPEG с качеством 25%

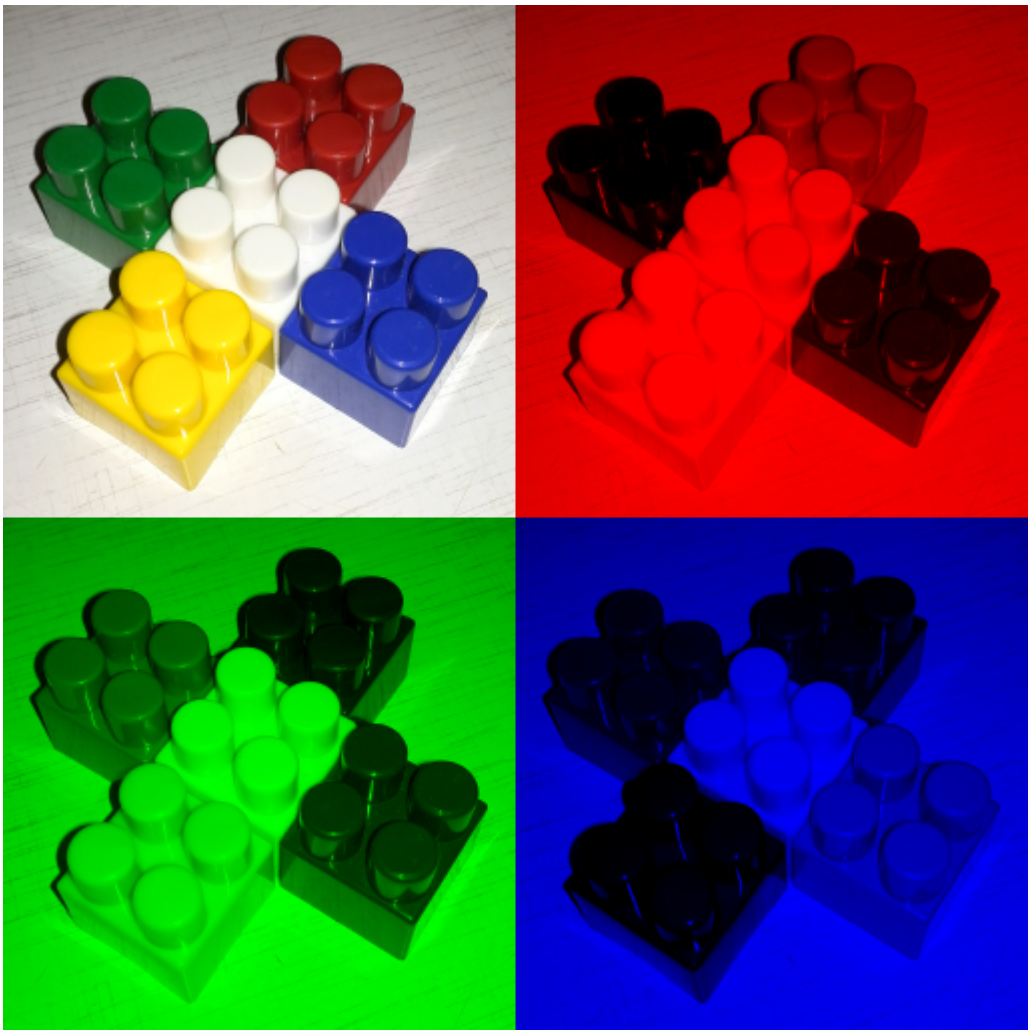

Рис. 2. Визуализация каналов исходного тестового изображения

Рис. 3. Визуализация каналов JPEG-версии тестового изображения

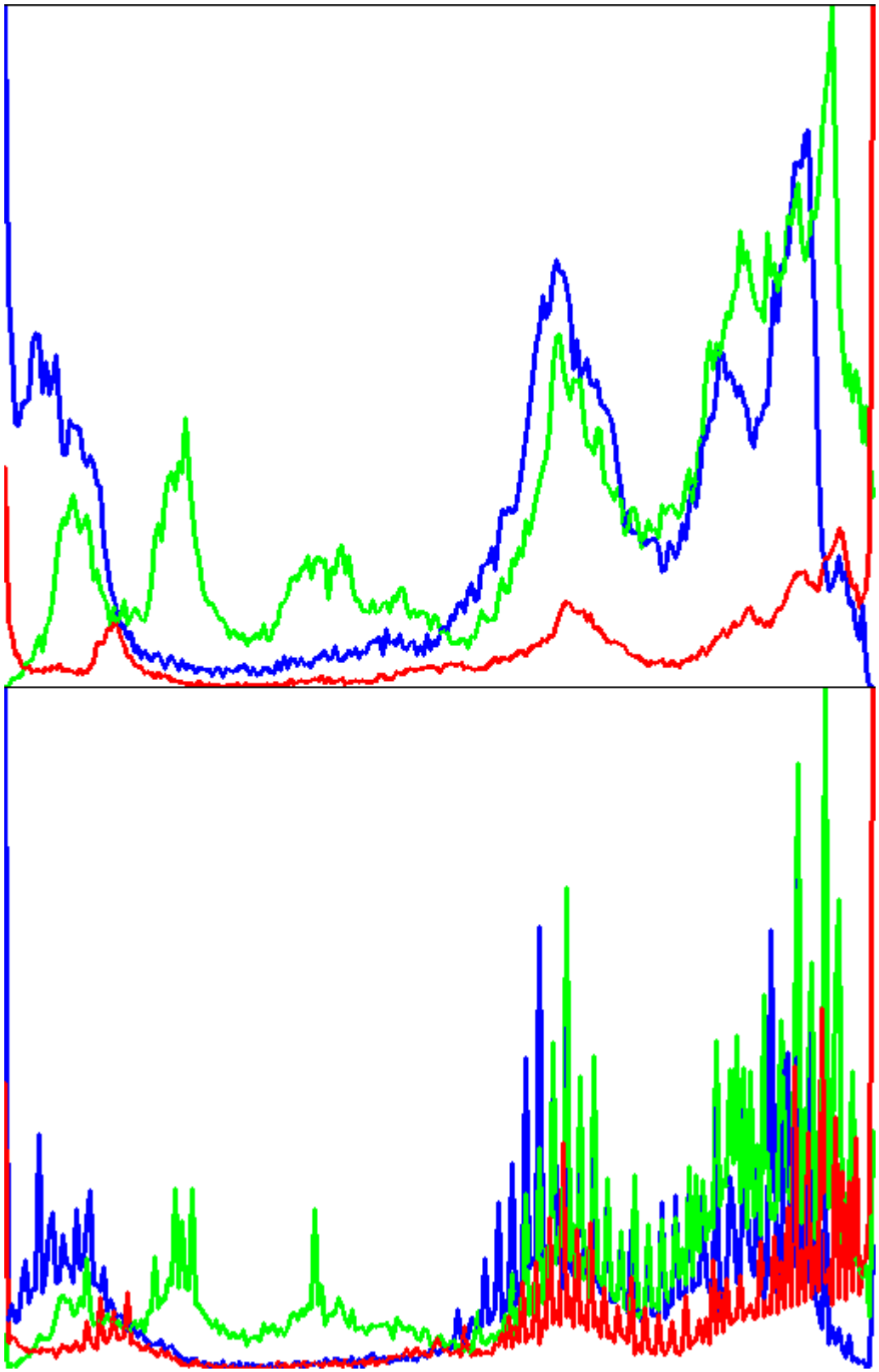

Рис. 4. Визуализация гистограм исходного и JPEG-версии тестового изображения

Текст программы

```
#include <opencv2/opencv.hpp>
#include <iostream>

#include <string>

cv::Mat split4(cv::Mat pic){
```

```
    cv::Mat channels[3];
    cv::split(pic, channels);

    cv::Mat zeros = cv::Mat::zeros(cv::Size(pic.cols, pic.rows), CV_8UC1);
    std::vector<cv::Mat> _blue = {channels[0], zeros, zeros};
    std::vector<cv::Mat> _green = {zeros, channels[1], zeros};
    std::vector<cv::Mat> _red = {zeros, zeros, channels[2]};
    cv::Mat Blue, Green, Red;
    cv::merge(_blue, Blue);
    cv::merge(_green, Green);
    cv::merge(_red, Red);

    cv::Mat bridge1, bridge2, res;
    cv::hconcat(pic, Red, bridge1);
    cv::hconcat(Green, Blue, bridge2);
    cv::vconcat(bridge1, bridge2, res);

    return res;
}

cv::Mat m_hist(cv::Mat pic) {
    std::vector<cv::Mat> bgr;
    cv::split(pic, bgr);

    int size = 256;

    float range[] = {0, 256};
    const float* histRange[] = {range};

    bool uniform = true, accumulate = false;
    cv::Mat blue, green, red;
    cv::calcHist(&bgr[0], 1, 0, cv::Mat(), blue, 1, &size, histRange,
uniform, accumulate);
    cv::calcHist(&bgr[1], 1, 0, cv::Mat(), green, 1, &size, histRange,
uniform, accumulate);
    cv::calcHist(&bgr[2], 1, 0, cv::Mat(), red, 1, &size, histRange,
uniform, accumulate);

    int width = 512, height = 400;
    int total = cvRound((double) width/size);

    cv::Mat res(height, width, CV_8UC3, cv::Scalar(255, 255, 255));

    cv::normalize(blue, blue, 0, res.rows, cv::NORM_MINMAX, -1,
cv::Mat());
    cv::normalize(green, green, 0, res.rows, cv::NORM_MINMAX, -1,
cv::Mat());
    cv::normalize(red, red, 0, res.rows, cv::NORM_MINMAX, -1, cv::Mat());

    for(int i = 1; i < size; i++){
        //blue
        cv::line(res, cv::Point(total*(i-1), height -
cvRound(blue.at<float>(i-1))),
        cv::Point(total*(i), height - cvRound(blue.at<float>(i))),
```

```
            cv::Scalar(255, 0, 0), 2, 8, 0);
            //green
            cv::line(res, cv::Point(total*(i-1), height -
cvRound(green.at<float>(i-1))),
            cv::Point(total*(i), height - cvRound(green.at<float>(i))),
            cv::Scalar(0, 255, 0), 2, 8, 0);
            //red
            cv::line(res, cv::Point(total*(i-1), height -
cvRound(red.at<float>(i-1))),
            cv::Point(total*(i), height - cvRound(red.at<float>(i))),
            cv::Scalar(0, 0, 255), 2, 8, 0);
        }

        return res;
}

int main() {

    // 1. В качестве тестового использовать изображение
data/cross_0256x0256.png

    std::string path_img =
cv::samples::findFile("../data/cross_0256x0256.png");
    cv::Mat img = cv::imread(path_img);

    if (img.empty()) {
        std::cout << "Could not read image!";
        return EXIT_FAILURE;
    }

    // 2. Сохранить тестовое изображение в формате JPEG с качеством 25%.

    cv::imwrite("cross_0256x0256_025.jpg", img, {
cv::IMWRITE_JPEG_QUALITY, 25 });

    // 3. Используя cv::merge и cv::split сделать "мозаику" с
визуализацией каналов для исходного тестового изображения и JPEG-версии
тестового изображения

    //PNG
    cv::imwrite("cross_0256x0256_png_channels.png", split4(img));

    //JPEG
    cv::Mat jpeg = cv::imread("cross_0256x0256_025.jpg",
cv::IMREAD_COLOR);
    cv::imwrite("cross_0256x0256_jpg_channels.png", split4(jpeg));

    // - левый верхний - трехканальное изображение
    // - левый нижний - монохромная (черно-зеленая) визуализация канала G
    // - правый верхний - монохромная (черно-красная) визуализация канала
R
    // - правый нижний - монохромная (черно-синяя) визуализация канала B
    // 4. Результы сохранить для вставки в отчет
```

```
    // 5. Сделать мозаику из визуализации гистограммы для исходного
тестового изображения и JPEG-версии тестового изображения, сохранить для
вставки в отчет.

    cv::Mat hist;
    cv::Mat line(1, 512, CV_8UC3, cv::Scalar(0, 0, 0));

    cv::vconcat(line, m_hist(img), hist);
    cv::vconcat(hist, line, hist);
    cv::vconcat(hist, m_hist(jpeg), hist);
    //cv::vconcat(hist, line, hist);
    cv::imwrite("cross_0256x0256_hists.png", hist);

    // cv::Mat pic1 = cv::imread("cross_0256x0256_png_channels.png",
cv::IMREAD_COLOR);
    // cv::Mat pic2 = cv::imread("cross_0256x0256_jpg_channels.png",
cv::IMREAD_COLOR);

    // cv::imshow("pic1", pic1);
    // cv::imshow("pic2", pic2);
    // cv::imshow("hist1", m_hist(pic1));
    // cv::imshow("hist2", m_hist(pic2));
    // cv::imshow("hist", hist);
    // cv::waitKey(0);
    // cv::destroyAllWindows();

    return 0;
}
```