# Работа 2. Исследование каналов и JPEG-сжатия

автор: Костромин А.Ю.
дата: 2022-02-23T17:35:53

url: https://github.com/alexanderkostromin/imagee-processing/tree/master/polevoy_d_v/prj.labs/lab02

## 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%

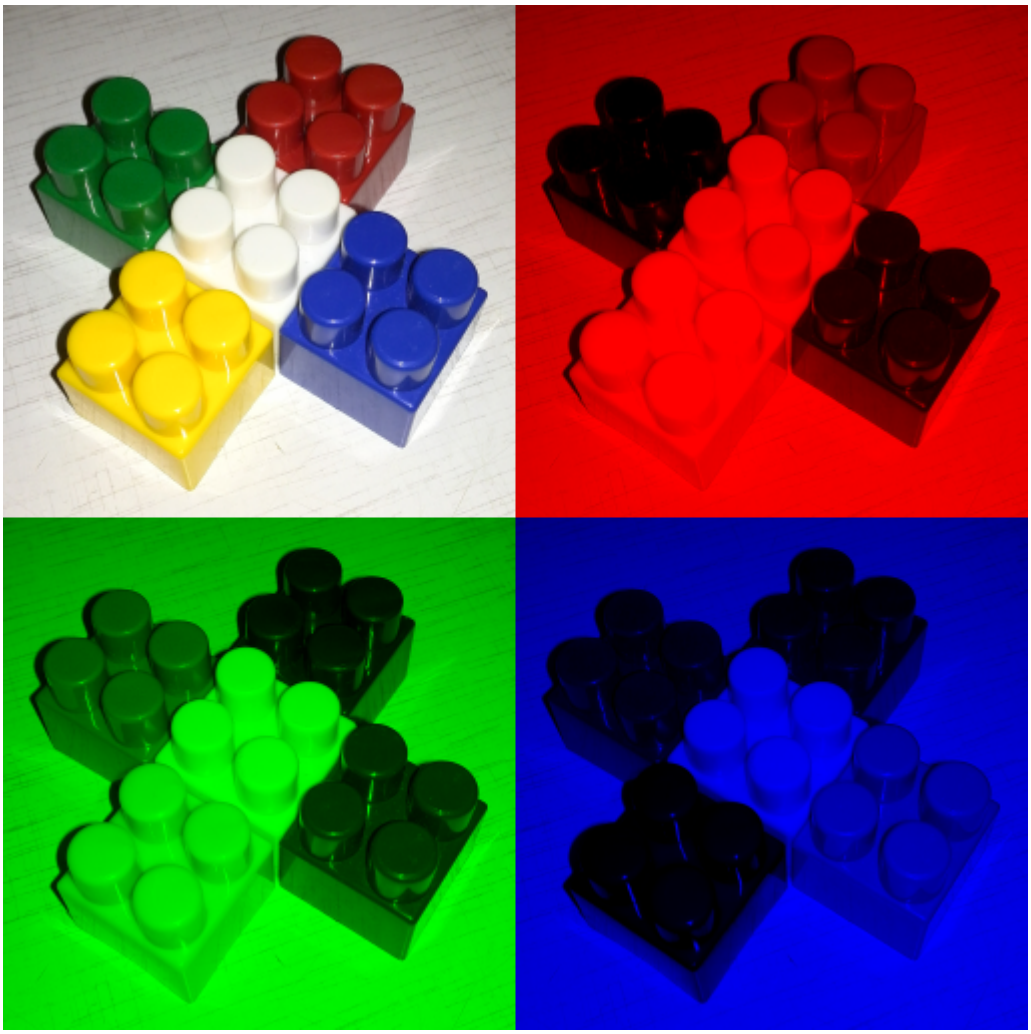

Рис. 2. Визуализация каналов исходного тестового изображения

Рис. 3. Визуализация каналов JPEG-версии тестового изображения

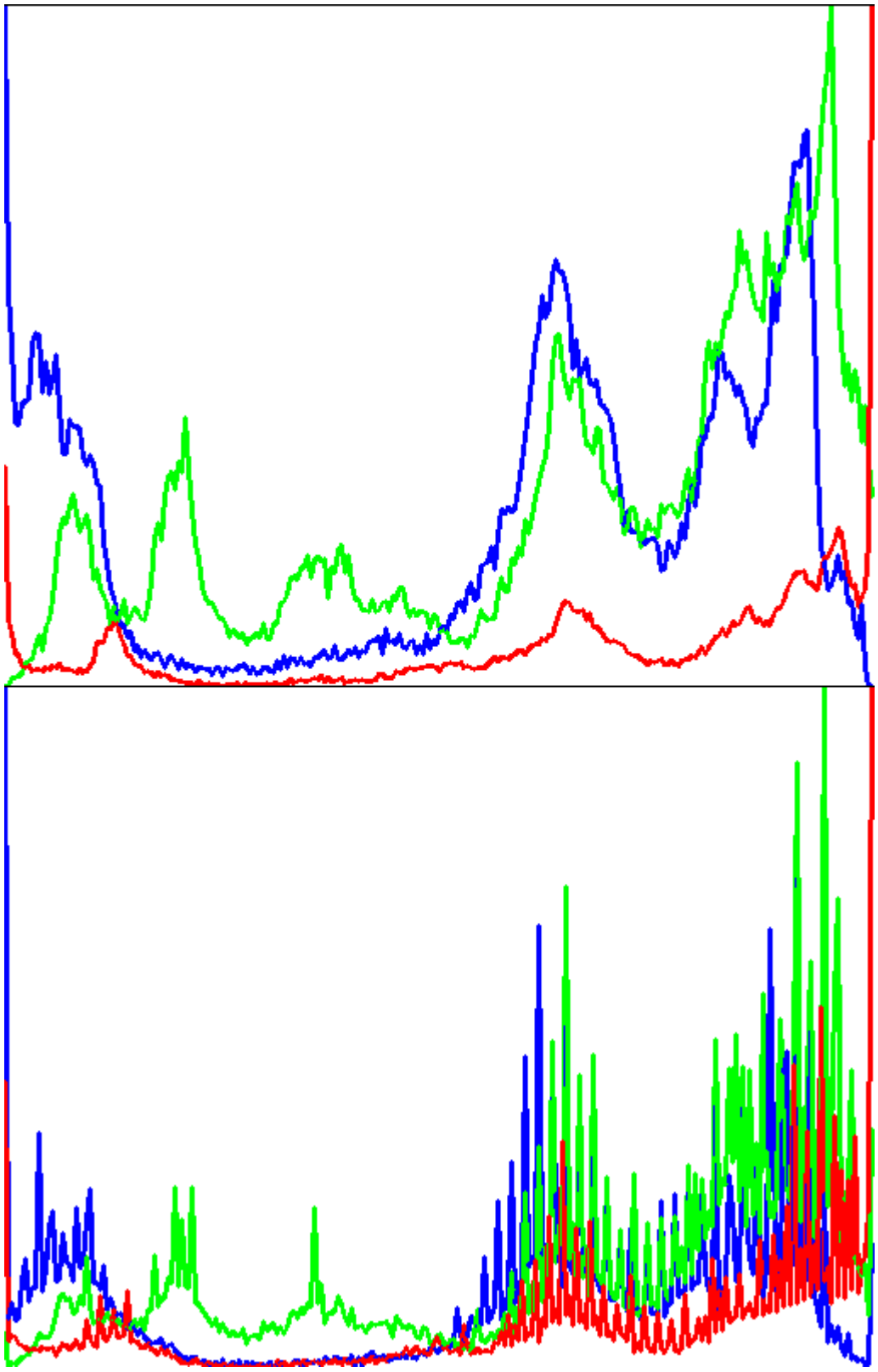

Рис. 4. Визуализация гистограм исходного и JPEG-версии тестового изображения

## Текст программы

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

cv::Mat do_split(cv::Mat src){
    cv::Mat channels[3];
    cv::split(src, channels);
```

```
    cv::Mat z = cv::Mat::zeros(cv::Size(src.cols, src.rows), CV_8UC1);
    std::vector<cv::Mat> channelB = {channels[0], z, z};
    std::vector<cv::Mat> channelG = {z, channels[1], z};
    std::vector<cv::Mat> channelR = {z, z, channels[2]};
    cv::Mat Blue, Green, Red;
    cv::merge(channelB, Blue);
    cv::merge(channelG, Green);
    cv::merge(channelR, Red);

    cv::Mat bridge1, bridge2, res;
    cv::hconcat(src, Red, part1);
    cv::hconcat(Green, Blue, part2);
    cv::vconcat(part1, part2, res);

    return res;
}

cv::Mat do_hist(cv::Mat src) {
    std::vector<cv::Mat> channels;
    cv::split( src, channels );
    int histSize = 256;
    float range[] = { 0, 256 };
    const float* histRange[] = { range };
    bool uniform = true, accumulate = false;
    cv::Mat b_hist, g_hist, r_hist;
    cv::calcHist( &channels[0], 1, 0, cv::Mat(), b_hist, 1, &histSize,
histRange, uniform, accumulate );
    cv::calcHist( &channels[1], 1, 0, cv::Mat(), g_hist, 1, &histSize,
histRange, uniform, accumulate );
    cv::calcHist( &channels[2], 1, 0, cv::Mat(), r_hist, 1, &histSize,
histRange, uniform, accumulate );
    int hist_w = 512, hist_h = 400;
    int bin_w = cvRound( (double) hist_w/histSize );
    cv::Mat histImage( hist_h, hist_w, CV_8UC3, cv::Scalar(255,255,255) );
    cv::normalize(b_hist, b_hist, 0, histImage.rows, cv::NORM_MINMAX, -1,
cv::Mat() );
    cv::normalize(g_hist, g_hist, 0, histImage.rows, cv::NORM_MINMAX, -1,
cv::Mat() );
    cv::normalize(r_hist, r_hist, 0, histImage.rows, cv::NORM_MINMAX, -1,
cv::Mat() );
    for( int i = 1; i < histSize; i++ )
    {
        line( histImage, cv::Point( bin_w*(i-1), hist_h -
cvRound(b_hist.at<float>(i-1)) ),
              cv::Point( bin_w*(i), hist_h - cvRound(b_hist.at<float>(i))
),
              cv::Scalar( 255, 0, 0), 2, 8, 0  );
        line( histImage, cv::Point( bin_w*(i-1), hist_h -
cvRound(g_hist.at<float>(i-1)) ),
              cv::Point( bin_w*(i), hist_h - cvRound(g_hist.at<float>(i))
),
              cv::Scalar( 0, 255, 0), 2, 8, 0  );
        line( histImage, cv::Point( bin_w*(i-1), hist_h -
cvRound(r_hist.at<float>(i-1)) ),
```

```
                cv::Point( bin_w*(i), hist_h - cvRound(r_hist.at<float>(i))
),
                cv::Scalar( 0, 0, 255), 2, 8, 0  );
    }
    return histImage;
}

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);

    // 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

    cv::imwrite("cross_0256x0256_png_channels.png", do_split(img));

    cv::Mat jpeg = cv::imread("cross_0256x0256_025.jpg",
cv::IMREAD_COLOR);
    cv::imwrite("cross_0256x0256_jpg_channels.png", do_split(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 ln(1, 512, CV_8UC3, cv::Scalar(0, 0, 0));

    cv::vconcat(ln, do_hist(img), hist);
    cv::vconcat(hist, ln, hist);
    cv::vconcat(hist, do_hist(jpeg), hist);
    cv::imwrite("cross_0256x0256_hists.png", hist);

    return 0;
}
```