Работа 2. Исследование каналов и JPEG-сжатия

автор: Егубова П.А.
дата: 2021-03-04T14:38:27

url: https://mysvn.ru/egubova_p_a/egubova_p_a/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%

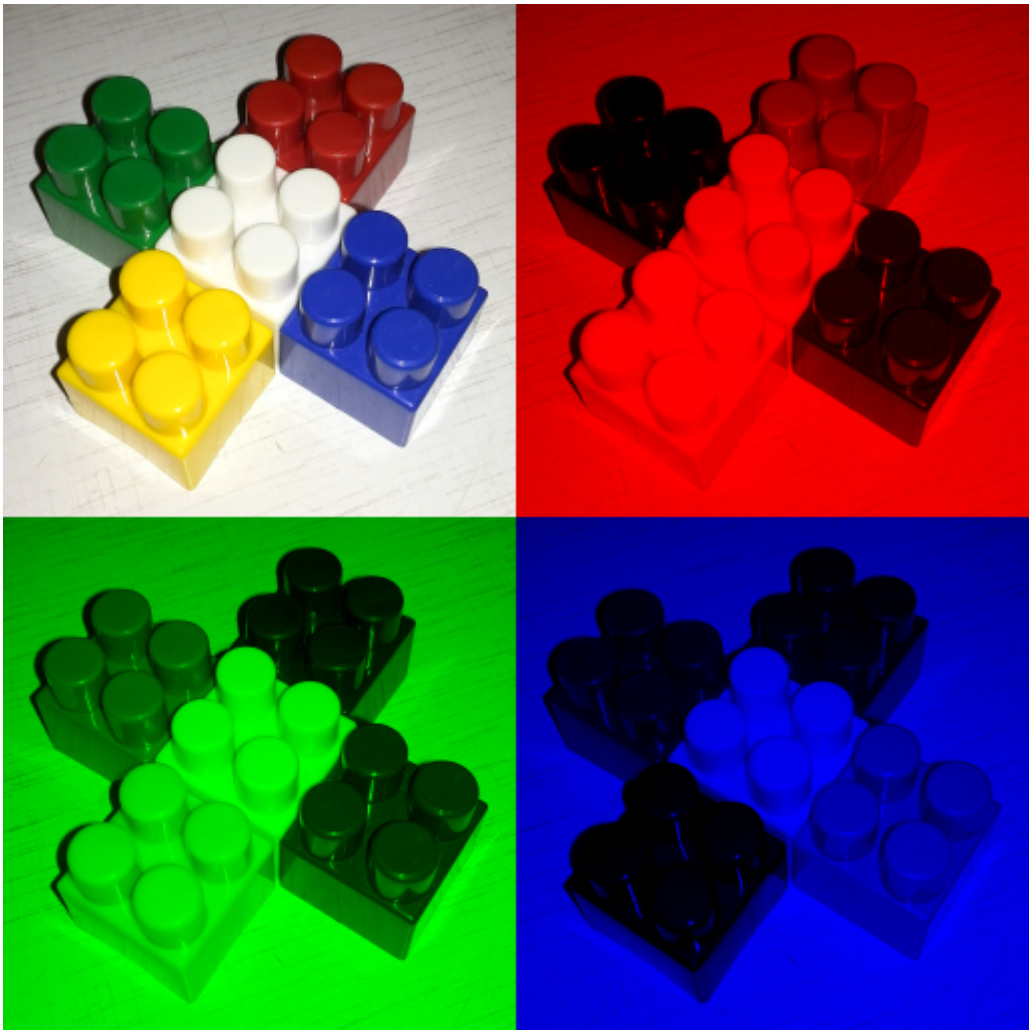

Рис. 2. Визуализация каналов исходного тестового изображения

Рис. 3. Визуализация каналов JPEG-версии тестового изображения

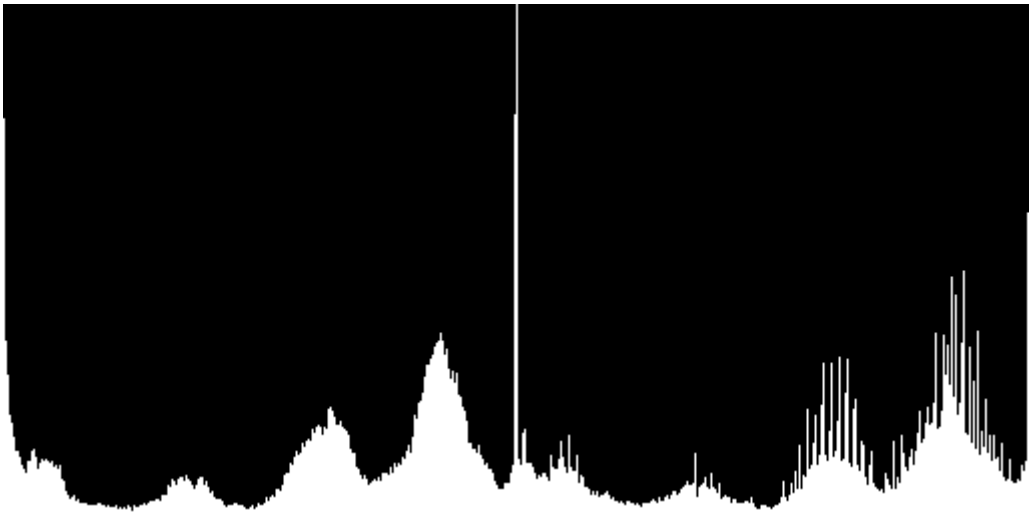

Рис. 3. Визуализация гистограм исходного и JPEG-версии тестового изображения

Текст программы

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

using namespace cv;
using namespace std;


Mat viz_channels(Mat& img) {
    Mat img_res(img.rows * 2, img.cols * 2, CV_8UC3);
```

```
    img.copyTo(img_res(Rect2d(0, 0, img.cols, img.rows)));

    vector<Mat> channels_img;
    for (int i = 0; i < 3; ++i) {
        Mat channels[3];
        split(img, channels);
        channels[(i + 1) % 3] = Mat::zeros(img.rows, img.cols, CV_8UC1);
        channels[(i + 2) % 3] = Mat::zeros(img.rows, img.cols, CV_8UC1);
        Mat img_copy(img.rows, img.cols, CV_8UC3);
        img.copyTo(img_copy(Rect2d(0, 0, img.cols, img.rows)));
        merge(channels, 3, img_copy);
        channels_img.emplace_back(img_copy);
    }

    img.copyTo(img_res(Rect2d(0, 0, img.cols, img.rows)));
    channels_img[0].copyTo(img_res(cv::Rect2d(img.cols, img.rows, img.cols,
img.rows)));
    channels_img[1].copyTo(img_res(cv::Rect2d(0, img.rows, img.cols,
img.rows)));
    channels_img[2].copyTo(img_res(cv::Rect2d(img.cols, 0, img.cols,
img.rows)));

    return img_res;
}


vector<Mat> make_hists(vector<Mat>& imgs) {
    vector<vector<int>> hists;
    vector<Mat> img_res;
    int max = 0;
    for (auto& img : imgs) {
        vector<int> hist(256, 0);
        for (int i = 0; i < img.rows; ++i) {
            for (int j = 0; j < img.cols; ++j) {
                ++hist[img.at<uchar>(i, j)];
            }
        }
        hists.push_back(hist);
        max = std::max(max, *max_element(hist.begin(), hist.end()));
    }
    for (auto& hist : hists) {
        for (size_t i = 0; i < hist.size(); ++i) {
            hist[i] = hist[i] * 256 / max;
        }
        Mat hist_img = Mat::zeros(256, 256, CV_8U);
        for (int i = 0; i < (int)hist.size(); ++i) {
            for (int j = 0; j < hist[i]; ++j) {
                hist_img.at<uchar>(255 - j, i) = 255;
            }
        }
        img_res.push_back(hist_img);
    }
    return img_res;
}


int main() {
    Mat img = imread("cross_0256x0256.png");
```

```
    Mat png_channels = viz_channels(img);
    imwrite("cross_0256x0256_png_channels.png", png_channels);

    imwrite("cross_0256x0256_025.jpg", img, { IMWRITE_JPEG_QUALITY, 25 });
    auto jpg_channels = viz_channels(imread("cross_0256x0256_025.jpg"));
    imwrite("cross_0256x0256_jpg_channels.png", jpg_channels);

    vector<Mat> imgs;
    imgs.push_back(img);
    imgs.push_back(imread("cross_0256x0256_025.jpg"));
    vector<Mat> hists = make_hists(imgs);
    Mat hists_img = Mat(256, 256 * 2, CV_8U);
    hists[0].copyTo(hists_img(Rect2d(0, 0, 256, 256)));
    hists[1].copyTo(hists_img(Rect2d(256, 0, 256, 256)));
    imwrite("cross_0256x0256_hists.png", hists_img);

    return 0;
}
```