Работа 2. Исследование каналов и JPEG-сжатия

автор: Машуров В. В. дата: 2022-02-21T18:07:12

url: GitHub - MVVladimir/mashurov_v_v

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%

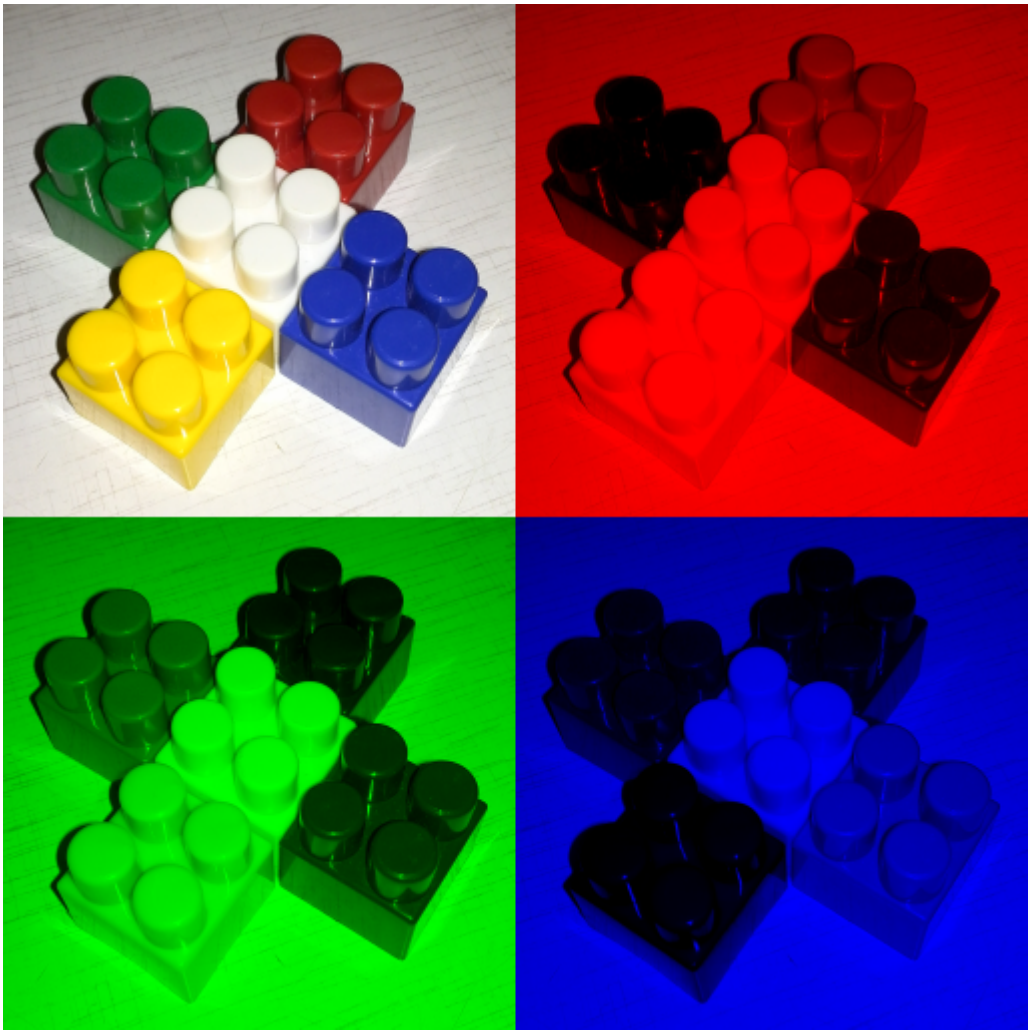

Рис. 2. Визуализация каналов исходного тестового изображения

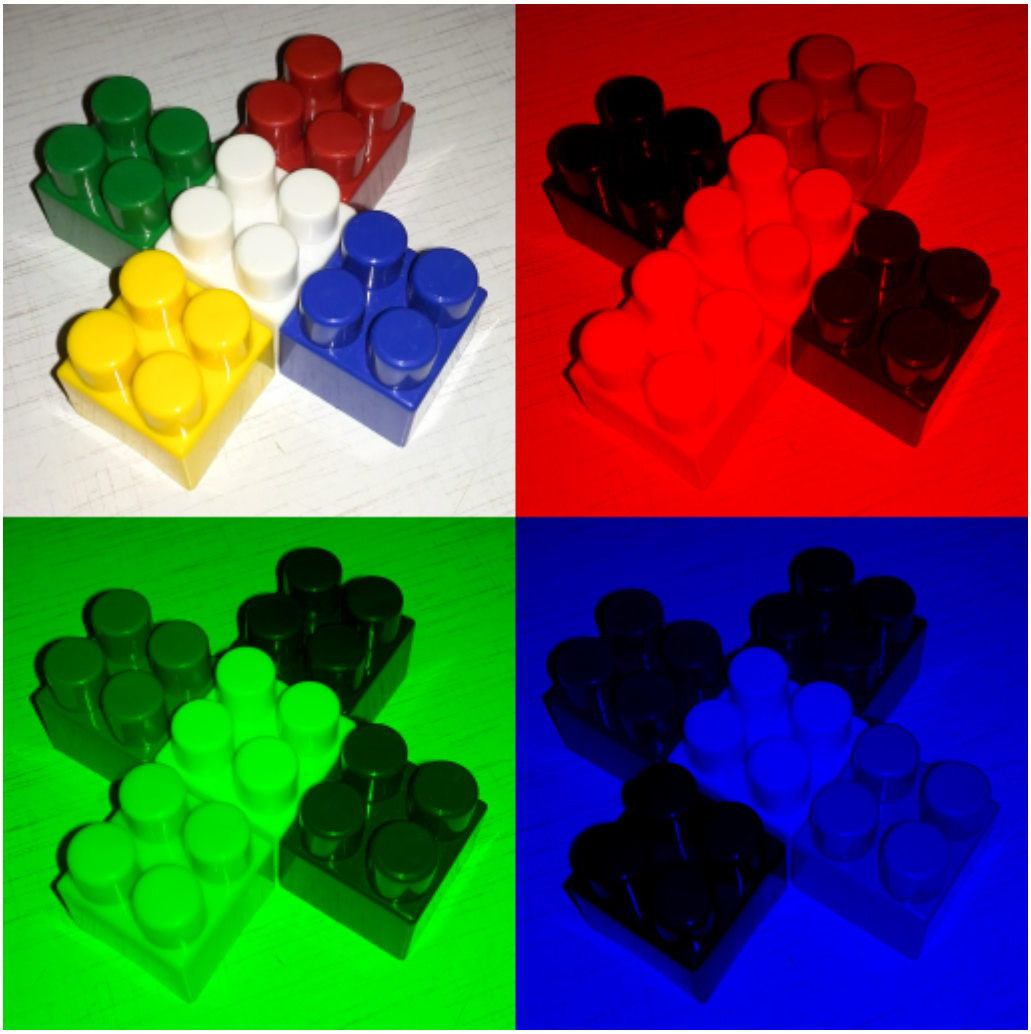

Рис. 3. Визуализация каналов JPEG-версии тестового изображения

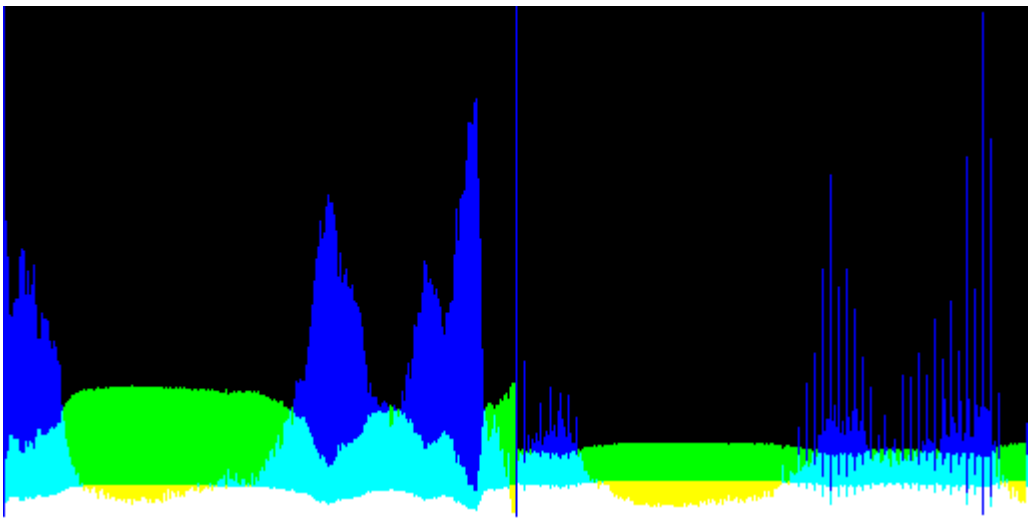

Рис. 3. Визуализация гистограм исходного и JPEG-версии тестового изображения

Текст программы

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

int max(const std::vector<int>& v) {

    int max = v[0];

    for (size_t i = 0; i < v.size(); i += 1)
        max = max < v[i] ? v[i] : max;

    return max;
}

void calcHist(std::vector<std::vector<int>>& hist, const cv::Mat& img) {
    for (std::ptrdiff_t i = 0; i < hist.size(); i += 1)
    {
        for (std::ptrdiff_t j = 0; j < hist[i].size(); j += 1)
        {
            hist[i][j] = 0;
        }
    }

    for (std::ptrdiff_t x = 0; x < img.cols; x += 1)
    {
        for (std::ptrdiff_t y = 0; y < img.rows; y += 1)
        {
            hist[0][img.at<cv::Vec3b>(x, y)[0]] += 1;
            hist[1][img.at<cv::Vec3b>(x, y)[1]] += 1;
            hist[2][img.at<cv::Vec3b>(x, y)[2]] += 1;
        }
    }

    int max_blue_color = max(hist[0]);
    int max_green_color = max(hist[1]);
    int max_red_color = max(hist[2]);

    for (std::ptrdiff_t i = 0; i < hist[0].size(); i += 1)
    {
        hist[0][i] = 255 - std::ceil(((double)hist[0][i] / (double)max_blue_color) *
255.);
        hist[1][i] = 255 - std::ceil(((double)hist[0][i] / (double)max_green_color) *
255.);
        hist[2][i] = 255 - std::ceil(((double)hist[0][i] / (double)max_red_color) *
255.);
    }

    for (std::ptrdiff_t i = 0; i < hist.size(); i += 1)
    {
        for (std::ptrdiff_t j = 0; j < hist[i].size(); j += 1)
        {
```

```
            std::cout << hist[i][j] << " ";
        }
        std::cout << '\n';
    }
}

void fillVizHistCol(cv::Mat& vizHist, std::ptrdiff_t const col, std::ptrdiff_t const
level) {
    for (std::ptrdiff_t i = level; i < vizHist.rows; i += 1)
    {
        vizHist.at<cv::uint8_t>(i, col) = 255;
    }
}

int main() {
    std::string filename = "../data/cross_0256x0256.png";
    cv::Mat img_png = cv::imread(filename, cv::IMREAD_COLOR);

    std::vector<int> compression_params;
    compression_params.push_back(cv::IMWRITE_JPEG_QUALITY);
    compression_params.push_back(25);

    //PICTURE_1
    imwrite("./cross_0256x0256_025.jpg", img_png, compression_params); // Save as jpg
with 75 persents loss


//=========================CALCULATING=AND=MERGING=CHANNELS=========================
=====

    std::vector<cv::Mat> channels(3);
    cv::split(img_png, channels); // split img to 3 channels - BGR

    cv::Mat zero = cv::Mat::zeros(channels[0].rows, channels[0].cols,
channels[0].type());

    std::vector<cv::Mat> red = { zero, zero, channels[2] };
    std::vector<cv::Mat> green = { zero, channels[1], zero };
    std::vector<cv::Mat> blue = { channels[0], zero, zero };

    cv::Mat redChannel, greenChannel, blueChannel;

    cv::merge(red, redChannel);
    cv::merge(green, greenChannel);
    cv::merge(blue, blueChannel);

    std::vector<cv::Mat> upper = { img_png, redChannel };
    std::vector<cv::Mat> bottom = { greenChannel, blueChannel };
    cv::Mat upperPic, bottomPic, img_channels;

    cv::hconcat(upper, upperPic);
```

```
    cv::hconcat(bottom, bottomPic);
    cv::vconcat(upperPic, bottomPic, img_channels);

    //PICTURE_2
    imwrite("./cross_0256x0256_png_channels.png", img_channels);
    //PICTURE_3
    imwrite("./cross_0256x0256_jpg_channels.png", img_channels);


//======================CALCULATING=HISTOGRAMS=======================================
=========

    std::string filename_jpg = "./cross_0256x0256_0256.jpg";
    cv::Mat img_jpg = cv::imread(filename_jpg, cv::IMREAD_COLOR);

    std::vector<std::vector<int>> hist_jpg(3, std::vector<int>(256));
    std::vector<std::vector<int>> hist_png(3, std::vector<int>(256));

    calcHist(hist_jpg, img_jpg);
    calcHist(hist_png, img_png);

    // Summary histogram
    cv::Mat vizHist(256, 256, CV_8UC3, cv::Vec3b(0, 0, 0));

    // Histogram for jpg image - vizHist_jpg - and for every channel BGR
    cv::Mat vizHist_jpg(256, 256, CV_8UC3, cv::Vec3b(0, 0, 0));
    std::vector<cv::Mat> vizHist_jpg_levels = { cv::Mat(256, 256, CV_8UC1) ,
        cv::Mat(256, 256, CV_8UC1) , cv::Mat(256, 256, CV_8UC1) };

    vizHist_jpg_levels[0] = 0;
    vizHist_jpg_levels[1] = 0;
    vizHist_jpg_levels[2] = 0;

    // Histogram for png image - vizHist_png - and for every channel BGR
    cv::Mat vizHist_png(256, 256, CV_8UC3, cv::Vec3b(255, 255, 255));
    std::vector<cv::Mat> vizHist_png_levels = { cv::Mat(256, 256, CV_8UC1) ,
        cv::Mat(256, 256, CV_8UC1) , cv::Mat(256, 256, CV_8UC1) };

    vizHist_png_levels[0] = 0;
    vizHist_png_levels[1] = 0;
    vizHist_png_levels[2] = 0;

    // Colors templates in BGR
    cv::Vec3b blue_color = { 255, 0, 0 };
    cv::Vec3b green_color = { 0, 255, 0 };
    cv::Vec3b red_color = { 0, 0, 255 };

    for (std::ptrdiff_t i = 0; i < vizHist_jpg.cols; i += 1)
    {
        fillVizHistCol(vizHist_jpg_levels[0], i, hist_jpg[0][i]);
        fillVizHistCol(vizHist_jpg_levels[1], i, hist_jpg[1][i]);
```

```
        fillVizHistCol(vizHist_jpg_levels[2], i, hist_jpg[2][i]);

        fillVizHistCol(vizHist_png_levels[0], i, hist_png[0][i]);
        fillVizHistCol(vizHist_png_levels[1], i, hist_png[1][i]);
        fillVizHistCol(vizHist_png_levels[2], i, hist_png[2][i]);
    }

    cv::merge(vizHist_jpg_levels, vizHist_jpg);
    cv::merge(vizHist_png_levels, vizHist_png);

    cv::hconcat(std::vector<cv::Mat> { vizHist_png, vizHist_jpg }, vizHist);

    //PICTURE_4
    imwrite("./cross_0256x0256_hists.png", vizHist);
}
```