Работа 2. Визуализация искажений jpeg-сжатия

автор: Мартынов Д.А. дата: 08.05.2020

Задание

Для исходного изображения (сохраненного без потерь) создать jpeg версии с двумя уровнями качества (например, 95 и 65). Вычислить и визуализировать на одной “мозаике” исходное изображение, результаты сжатия, поканальные и яркостные различия.

Результаты

Рис 1. Исходное изображение

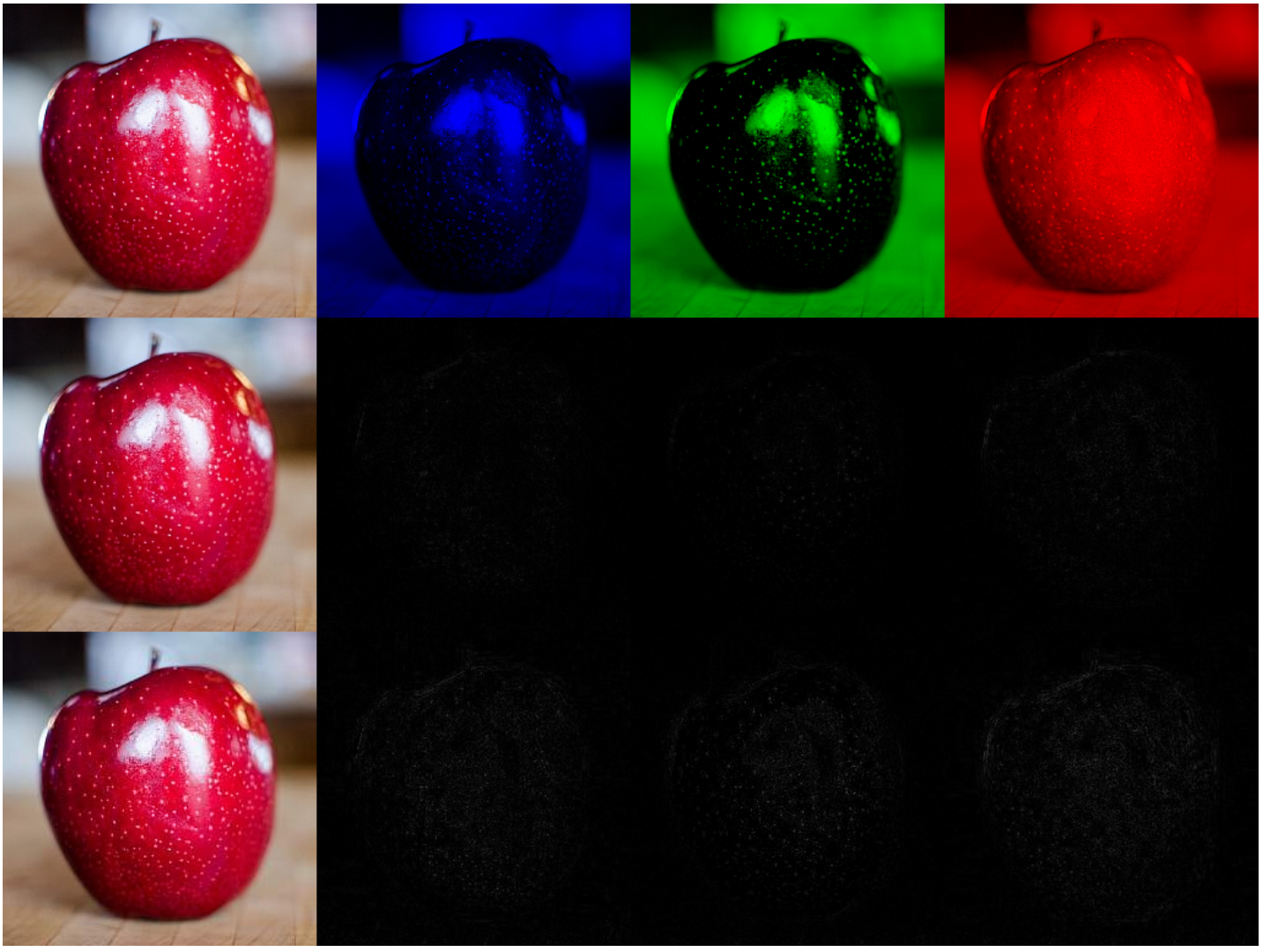

Рис 2. Результаты работы программы

Текст программы

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

std::vector<cv::Mat> absDifference(const std::vector<cv::Mat>& img, const std::vector<cv::Mat>& rgb)
{
```

```
    std::vector<cv::Mat> d;
    for (int i{ 0 }; i < 3; ++i)
    {
        cv::Mat temp;
        cv::absdiff(img[i], rgb[i], temp);
        d.push_back(temp);
    }
    return d;
}

int main()
{
    // Директория откуда читать оригинальные картинки
    const std::string readFrom{ "C:/Admin/Programming/C++/OpenCV/data/test_data/" };
    // Директория куда записывать
    const std::string writeTo{ "C:/Admin/Programming/C++/OpenCV/data/lab_data/" };

    cv::Mat img = cv::imread(readFrom + "apple_256x256.png", CV_LOAD_IMAGE_COLOR);

    // Запись изображения в формате jpeg(95) и jpeg(65)
    cv::imwrite(writeTo + "apple_256x256_95.jpg", img, { CV_IMWRITE_JPEG_QUALITY, 95 });
    cv::imwrite(writeTo + "apple_256x256_65.jpg", img, { CV_IMWRITE_JPEG_QUALITY, 65 });

    cv::Mat compressed95 = cv::imread("C:/Admin/Programming/C++/OpenCV/data/lab_data/apple_256x256_95.jpg", CV_LOAD_IMAGE_COLOR);
    cv::Mat compressed65 = cv::imread("C:/Admin/Programming/C++/OpenCV/data/lab_data/apple_256x256_65.jpg", CV_LOAD_IMAGE_COLOR);

    // Создание окна под все картинки
    int cx{ img.cols };
    int cy{ img.rows };
    cv::Mat allImages(cy * 3, cx * 4, img.type());

    // Выделение областей окна под мозаику
    std::vector<cv::Mat>split;
    for (int i{ 0 }; i < 3; ++i)
    {
        for (int j{ 0 }; j < 4; ++j)
            split.push_back(cv::Mat(allImages, cv::Rect2d(j * cx, i * cy, cx, cy)));
    }

    // Разделение картинок на BGR каналы
    std::vector<cv::Mat> rgb;
    std::vector<cv::Mat> rgb_95;
    std::vector<cv::Mat> rgb_65;
    cv::split(img, rgb);
    cv::split(compressed95, rgb_95);
    cv::split(compressed65, rgb_65);

    // Создал вектор всех картинок, чтобы потом в цикле просто скопировать их в области мозаики
    cv::Mat zeros(cv::Mat::zeros(cx, cy, CV_8UC1));
    std::vector<std::vector<cv::Mat>> tempV;

    tempV.push_back({ rgb[0], rgb[1],rgb[2] });
    tempV.push_back({ rgb[0], zeros, zeros });
    tempV.push_back({ zeros, rgb[1], zeros });
    tempV.push_back({ zeros, zeros, rgb[2] });

    // Разница по каналам с jpeg (95)
    std::vector<cv::Mat> temp = absDifference(rgb, rgb_95);
    tempV.push_back({ rgb_95[0], rgb_95[1],rgb_95[2] });
    tempV.push_back({ temp[0], temp[0] , temp[0] });
    tempV.push_back({ temp[1], temp[1] , temp[1] });
    tempV.push_back({ temp[2], temp[2] , temp[2] });

    // Разница по каналам с jpeg (65)
    temp = absDifference(rgb, rgb_65);
    tempV.push_back({ rgb_65[0], rgb_65[1],  rgb_65[2] });
    tempV.push_back({ temp[0], temp[0] , temp[0] });
    tempV.push_back({ temp[1], temp[1] , temp[1] });
    tempV.push_back({ temp[2], temp[2] , temp[2] });

    // Копирование картинок в выделенные области
    for (int i{ 0 }; i < 12; ++i)
    {
        cv::merge(tempV[i], split[i]);
    }

    cv::imwrite(writeTo + "lab2_all_images.png", allImages, { CV_IMWRITE_PNG_COMPRESSION, 0 });

    return 0;
}
```