Работа 3. Яркостные преобразования

автор: Хонер П.Д.
дата: 2022-03-20T13:08:00

https://github.com/pavelkhoner/OpenCV/tree/master/khoner_p_d/prj.labs/lab03

Задание

1. В качестве тестового использовать изображение data/cross_0256x0256.png
2. Сгенерировать нетривиальную новую функцию преобразования яркости (не стоит использовать слишком простые и слишком простые функции).
3. Сгенерировать визуализацию функцию преобразования яркости в виде изображения размером 512x512, черные точки а белом фоне.
4. Преобразовать пиксели grayscale версии тестового изображения при помощи LUT для сгенерированной функции преобразования.
5. Преобразовать пиксели каждого канала тестового изображения при помощи LUT для сгенерированной функции преобразования.
6. Результы сохранить для вставки в отчет.

Результаты

Рис. 1. Исходное тестовое изображение

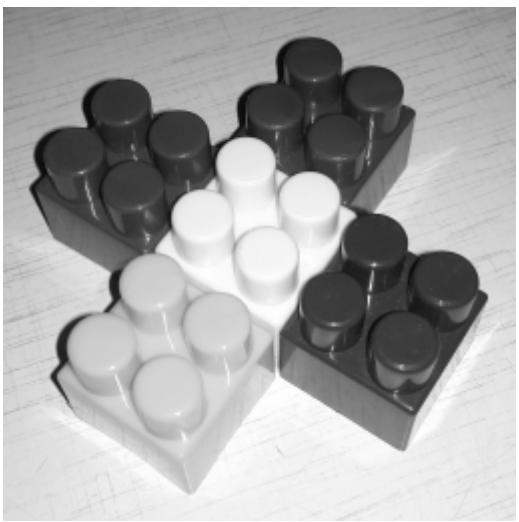

Рис. 2. Тестовое изображение greyscale

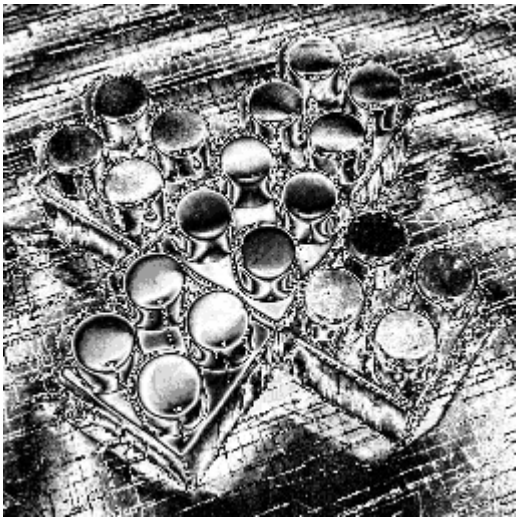

Рис. 3. Результат применения функции преобразования яркости для greyscale

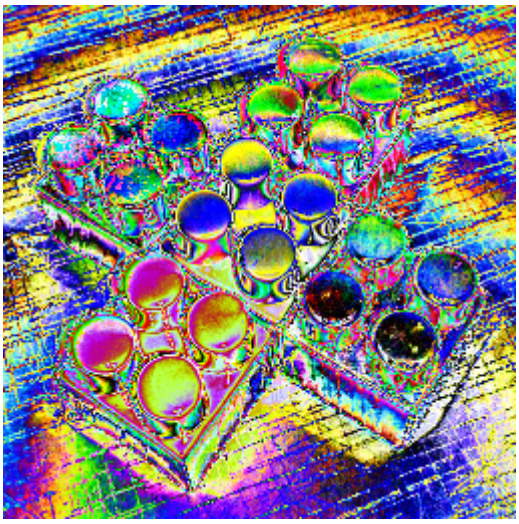

Рис. 4. Результат применения функции преобразования яркости для каналов

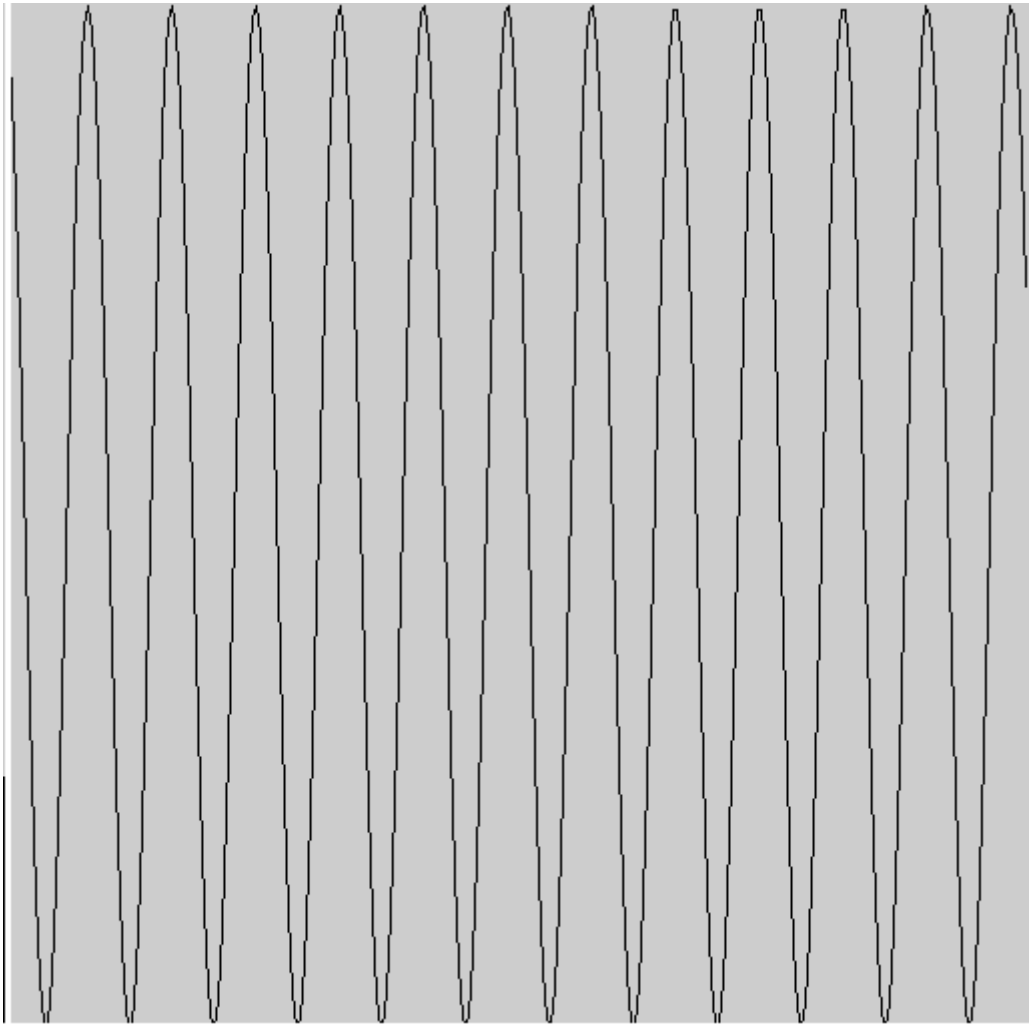

Рис. 5. Визуализация функции яркостного преобразования

Текст программы

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

int main() {
    //ini
    cv::Mat srcGreyImg =
cv::imread("C:/sandbox/khoner_p_d/data/cross_0256x0256.png",
cv::IMREAD_GRAYSCALE);
    cv::Mat srcRgbImg =
cv::imread("C:/sandbox/khoner_p_d/data/cross_0256x0256.png");

    cv::imwrite("lab03_rgb.png", srcRgbImg);
    cv::imwrite("lab03_gre.png", srcGreyImg);

    cv::Mat LUP(1, 256, CV_8UC1);

    uchar* m = LUP.ptr();

    for (int x = 0; x < 256; ++x) {
        m[x] = 128 * (cos(x * 0.3) + 5);
    }

    cv::Mat check(180, 768, CV_8UC1);
    check = 0;
```

```
    for (int y = 0; y < 180; ++y)
        for (int x = 0; x < 768; ++x)
            check.at<uchar>(y, x) = x / 3;

    cv::Mat iCheck;
    cv::LUT(check, LUP, iCheck);
    cv::imwrite("lab03_check.png", iCheck);

    cv::Mat img;
    cv::LUT(srcRgbImg, LUP, img);
    cv::imwrite("lab03_rgb_res.png", img);

    cv::LUT(srcGreyImg, LUP, img);
    cv::imwrite("lab03_gre_res.png", img);

    cv::Mat srcFunction(512, 512, CV_8UC1);

    cv::Point p1(0, 510), p2(512, 510);
    cv::Point p3(2, 0), p4(2, 512);
    int T = 2;

    for (int i = 1; i < 256; i++) {
        cv::line(srcFunction,
            cv::Point(i - 1 << 1, 511 - (((int)m[i - 1]) << 1)),
            cv::Point(i << 1, 511 - (((int)m[i]) << 1)),
            0, 1, 0);
    }

    cv::line(srcFunction, p1, p2, cv::Scalar(255, 0, 0), T, cv::LINE_4);
    cv::line(srcFunction, p3, p4, cv::Scalar(255, 0, 0), T, cv::LINE_4);

    cv::imwrite("lab03_viz_func.png", srcFunction);
}
```