Работа 3. Яркостные преобразования изображений

автор: Дворядкин К.А. дата: 2022-03-01T14:05:58

url: https://github.com/DiorChoppa/imageprocessing-spring-2022/tree/main/prj.labs/lab03

Задание

1. В качестве тестового использовать изображение data/cross_0256x0256.png
2. Сгенерировать нетривиальную новую функцию преобразования яркости (не стоит использовать линейную функцию, гамму, случайная).
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

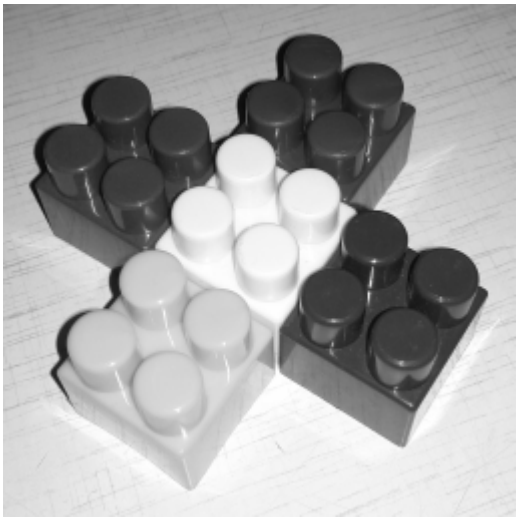

Рис. 2. Тестовое изображение greyscale

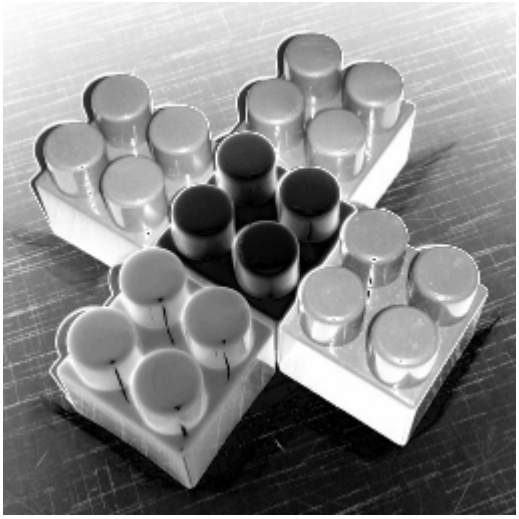

Рис. 3. Результат применения функции преобразования яркости для greyscale

Рис. 4. Результат применения функции преобразования яркости для каналов

Рис. 5. Проверка функии преоразования на градиенте из lab01

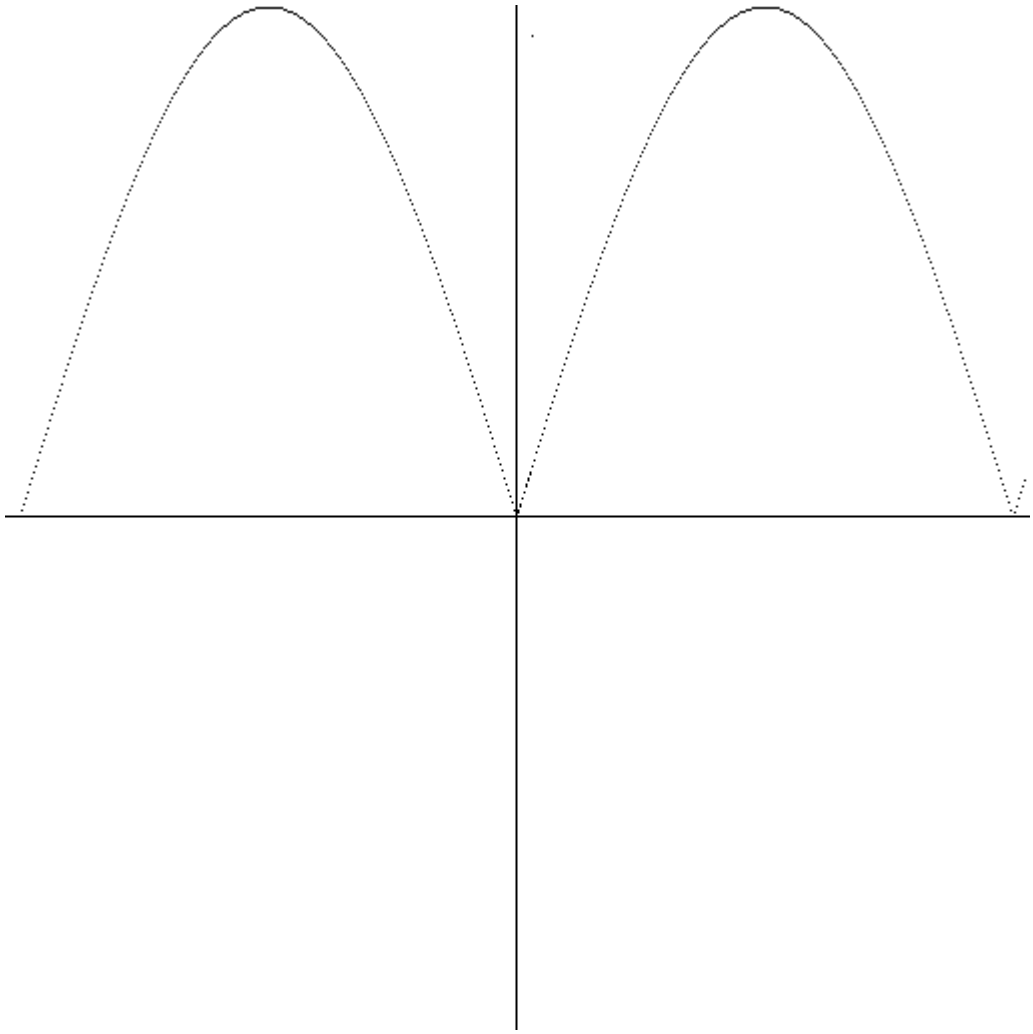

Рис. 6. Визуализация функции яркостного преобразования

Текст программы

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

// 2. Сгенерировать нетривиальную новую функцию преобразования яркости (не
стоит использовать линейную функцию, гамму, случайная).

double ft_brchange(int pixel) {
    return abs(sin(pixel/79.0))*255;
}

int main() {

    // Creating a gradient

    cv::Mat gradient(180, 768, CV_8UC1);
    for (int i = 0; i < gradient.cols; i++)
    {
        for (int j = 0; j < gradient.rows; j++)
        {
            gradient.at<uint8_t>(j, i) = i / 3;
        }
```

```
    }

    for (int i = 0; i < gradient.cols; i++)
    {
        for (int j = 0; j < gradient.rows; j++)
        {
            gradient.at<uint8_t>(j, i) = ft_brchange(i/3);
        }

    }


    // 1. В качестве тестового использовать изображение
data/cross_0256x0256.png

    std::string path_img =
cv::samples::findFile("../../data/lab03_rgb.png");
    cv::Mat img = cv::imread(path_img);

    if (img.empty()) {
        std::cout << "Could not read image!";
        return EXIT_FAILURE;
    }

    cv::imwrite("lab03_rgb.png", img);


    // 3. Сгенерировать визуализацию функцию преобразования яркости в виде
изображения размером 512x512, черные точки а белом фоне.

    cv::Mat lut(1, 256, CV_8UC1);
    for(int i = 0; i < 256; i++){
        lut.at<uchar>(0, i) = static_cast<uint>(ft_brchange(i));
    }

    cv::Mat clear(512, 512, CV_8UC1, 255);
    for(int i = 0; i < 512; i++){
        if (i < 255){
            clear.at<uchar>(256 - lut.at<uchar>(0, i) - 1, i+255) = 0;
        }
        else {
            clear.at<uchar>(255 - lut.at<uchar>(0, i-255) - 1, i+264) = 0;
        }
    }

    for (int i = 0; i < 512; i++){
        clear.at<uint8_t>(255, i) = 0;
        clear.at<uint8_t>(i, 255) = 0;
    }
    cv::imwrite("lab03_viz_func.png", clear);


    // 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.

    cv::Mat grayscale;
    cv::cvtColor(img, grayscale, cv::COLOR_BGR2GRAY);
    cv::imwrite("lab03_gre.png", grayscale);

    cv::Mat res_img, res_gray;
    cv::LUT(img, lut, res_img);
    cv::LUT(grayscale, lut, res_gray);

    cv::imwrite("lab03_gre_res.png", res_gray);
    cv::imwrite("lab03_rgb_res.png", res_img);
    cv::imwrite("gradient.png", gradient);


    return 0;
}
```