

## Работа 3. Яркостные преобразования изображений

автор: Костромин А.Ю. дата: 2022-03-14T04:05:24

url: [GitHub](#)

### 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. Результаты 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. Отношение старой яркости к новой.



Рис. 4. Результат применения функции преобразования яркости для каналов



Рис. 5. Проверка функции преобразования на градиенте из lab01

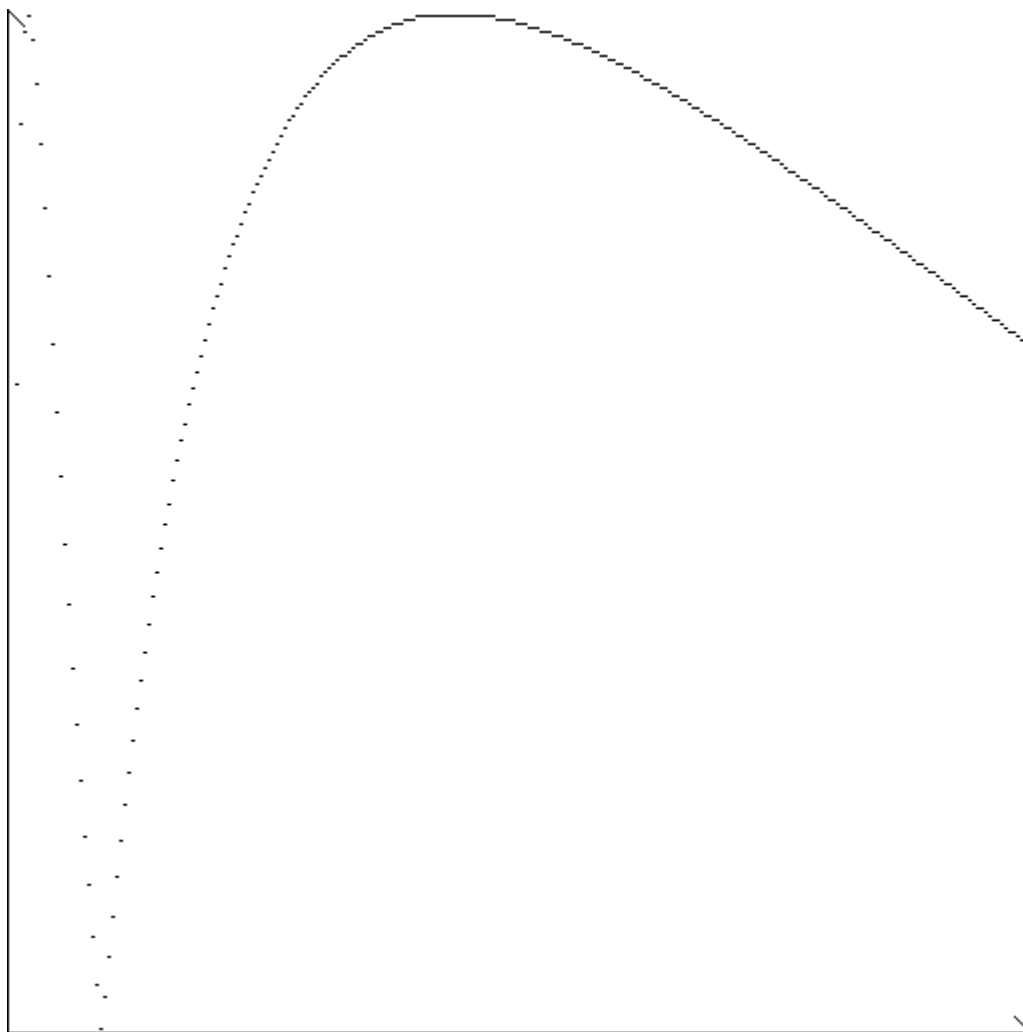


Рис. 6. Визуализация функции яркостного преобразования

### 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 p) {
    return abs(sin(log(p)))*255.0;
}

int main() {

    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("/Users/blssd/image_processing/polevoy_d_v/prj.labs/
lab03/lab03_rgb.png");
cv::Mat img = cv::imread(path_img);
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<uint8_t>(0, i) = static_cast<uint8_t>(ft_brchange(i));
}

cv::Mat ft(512, 512, CV_8UC1, 255);
for(int i = 0; i < 512; i++){
    ft.at<uchar>(512 - 2*lut.at<uchar>(0, i/2) - 1, i) = 0;
}
for (int i = 0; i < 512; i++){
    ft.at<uint8_t>(511, i) = 0;
    ft.at<uint8_t>(i, 0) = 0;
}
for (int i = 0; i <= 8; i++){
    ft.at<uint8_t>(i, i) = 0;
    ft.at<uint8_t>(511-i, 511-i) = 0;
}

cv::imwrite("lab03_viz_func.png", ft);

// 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 greyscale;
cv::cvtColor(img, greyscale, cv::COLOR_BGR2GRAY);
cv::imwrite("lab03_gre.png", greyscale);
cv::Mat res_img, res_grey;
```

```
cv::LUT(img, lut, res_img);
cv::LUT(greyscale, lut, res_grey);
cv::imwrite("lab03_gre_res.png", res_grey);
cv::imwrite("lab03_rgb_res.png", res_img);
cv::imwrite("gradient.png", gradient);

cv::imshow("grey", res_grey);
cv::imshow("img", res_img);
cv::imshow("gradient", gradient);
cv::imshow("ft", ft);
cv::waitKey();

return 0;
}
```