

# Работа 1. Исследование гамма-коррекции

---

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

url: [GitHub - MVVladimir/mashurov\\_v\\_v](#)

## Задание

1. Сгенерировать серое тестовое изображение  $I_1$  в виде прямоугольника размером 768x60 пикселя с плавным изменением пикселей от черного к белому, одна градация серого занимает 3 пикселя по горизонтали.
2. Применить к изображению  $I_1$  гамма-коррекцию с коэффициентом из интервала 2.2-2.4 и получить изображение  $G_1$  при помощи функции `pow`.
3. Применить к изображению  $I_1$  гамма-коррекцию с коэффициентом из интервала 2.2-2.4 и получить изображение  $G_2$  при помощи прямого обращения к пикселям.
4. Показать визуализацию результатов в виде одного изображения (сверху вниз  $I_1$ ,  $G_1$ ,  $G_2$ ).
5. Сделать замер времени обработки изображений в п.2 и п.3, результаты отфиксировать в отчете.

## Время выполнения

Замеры результатов выполнения проводились на Release версии сборки.

gamma-correction cv using open cv method: 1.8184 ms gamma-correction cv using for-pixel-by-pixel-cicle: 0.971 ms

## Результаты



Рис. 1. Результаты работы программы (сверху вниз  $I_1$ ,  $G_1$ ,  $G_2$ )

## Текст программы

```

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

int main() {
    cv::Mat I_1(60, 768, CV_8UC1);
    cv::Mat G_1(60, 768, CV_8UC1);
    cv::Mat G_2(60, 768, CV_8UC1);

    cv::TickMeter I_1_tm, G_1_tm, G_2_tm;

    // draw dummy image
    I_1 = 0;

    I_1_tm.start();

    for (int i = 0; i < I_1.rows; i++)
        for (int j = 0; j < I_1.cols; j++) {
            I_1.at<uint8_t>(i, j) += (j / 3);
        }

    I_1_tm.stop();

    G_1 = I_1.clone();
    G_2 = I_1.clone();

    cv::Mat G_1_float;

    G_1_tm.start();

    G_1.convertTo(G_1_float, CV_32FC1, 1./255.);
    cv::pow(G_1_float, 2.3, G_1_float);
    G_1_float.convertTo(G_1, CV_8UC1, 255);

    G_1_tm.stop();

    G_2_tm.start();

    for (int i = 0; i < G_2.rows; i++)
        for (int j = 0; j < G_2.cols; j++) {
            G_2.at<uint8_t>(i, j) = std::pow(G_2.at<uint8_t>(i, j) / 255., 2.3) * 255.;
        }

    G_2_tm.stop();

    cv::Mat matArray[] = { I_1,
                           G_1,
                           G_2, };

    cv::Mat out;
    cv::vconcat(matArray, 3, out);
}

```

```
std::cout << "\n gamma-correction cv using open cv method: " <<
G_1_tm.getTimeMilli()
    << "\n gamma-correction cv using for pixel by pixel cicle: " <<
G_2_tm.getTimeMilli() << std::endl;

// save result
cv::imwrite("I_1.png", I_1);
cv::imwrite("G_1.png", G_1);
cv::imwrite("G_2.png", G_2);
cv::imwrite("lab01.png", out);
}
```