Работа 1. Исследование гамма-коррекции

автор: Балаев А. А.
дата: 14.02.2022

Задание

1. Сгенерировать серое тестовое изображение I_1 в виде прямоугольника размером 768x60 пикселя с плавным изменение 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 функци 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.

Результаты

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

Время обработки изображения в п.2

33748 микросекунд

Время обработки изображения в п.3

6328 микросекунд

Текст программы

```
#include <opencv2/opencv.hpp>
#include <chrono>
#include <iostream>

using namespace std::chrono;

int main() {

    const double Gamma1 = 2.2;
    const double Gamma2 = 2.3;

    cv::Mat img(180, 768, CV_8UC1);
    // draw dummy image
```

```
    img = 0;

    cv::Rect2d rc = { 0, 0, 768, 60 };

    for (ptrdiff_t y = 0; y < 180; ++y) {
        for (ptrdiff_t x = 0; x < 768; ++x) {
                img.at<uchar>(y, x) = x / 3;
        }
    }

    //cv::rectangle(img, rc, { 150 }, 1);
    rc.y += rc.height;
    auto start_time1 = high_resolution_clock::now();

    cv::Mat new_image;
    img.copyTo(new_image);
    new_image.convertTo(new_image, CV_64FC1, 1.0f / 255.0f);
    cv::pow(new_image, Gamma1, new_image);
    new_image.convertTo(new_image, CV_8UC1, 255.0f);
    new_image(rc).copyTo(img(rc));

    auto finish_time1 = high_resolution_clock::now();
    auto duration1 = duration_cast<microseconds>(finish_time1 - start_time1);
    std::cout << duration1.count() << " microsecond(s)" << "\n";


    //cv::rectangle(img, rc, { 150 }, 1);
    rc.y += rc.height;

    auto start_time2 = high_resolution_clock::now();

    for (ptrdiff_t y = 120; y < 180; ++y) {
        for (ptrdiff_t x = 0; x < 768; ++x) {
            img.at<uchar>(y,x) = cv::saturate_cast<uchar>(cv::pow((img.at<uchar>
(y,x) / 255.0f), Gamma2) * 255.0f);
        }
    }
    auto finish_time2 = high_resolution_clock::now();
    auto duration2 = duration_cast<microseconds>(finish_time2 - start_time2);
    std::cout << duration2.count() << " microsecond(s)" << "\n";

    //cv::rectangle(img, rc, { 150 }, 1);
    //
    // save result

    cv::imwrite("lab01.png", img);

    cv::imshow("Output", img);

    cv::waitKey(0);
    return 0;
}
```