Работа 1. Исследование гамма-коррекции

автор: Конев Т. В.
дата: 2022-02-21T19:16:39

Github: https://github.com/timakon/KonevCV/tree/main/prj.labs/lab01

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)

После замера времени мы выяснили, что G_1 (3.6018 миллисекунд) работает быстрее чем G_2 (6.2128 миллисекунд).

Текст программы

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


int main() {
    cv::Mat I_1(60, 768, CV_8UC1);
    I_1 = 0;
    int color = 0;
    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;
        }
    }

    cv::Mat G_1 = I_1.clone();
    I_1.convertTo(G_1,CV_32FC1,1.0/255.0);
    pow(G_1, 2.2, G_1);
    G_1.convertTo(G_1, CV_8UC1, 255.0);

    cv::Mat G_2 = I_1.clone();
    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) / 255.0, 2.2) *
255;
        }
    }

    cv::Mat lab01;
    cv::Mat images[] = { I_1, G_1, G_2 };
    cv::vconcat(images,3, lab01);

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

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