Работа 1. Исследование гамма-коррекции

автор: Лоев В.А. дата:

https://mysvn.ru/LOEV V A/loev v a/prj.labs/lab 1/

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 прямого обращения к пикселям.
- 3. Применить к изображению I_1 гамма-коррекцию с коэффициентом из интервала 2.2-2.4 и получить изображение G_2 при помощи функции pow.
- 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

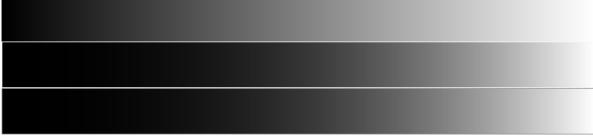


Рис. 1. Результаты работы программы (сверху вниз I_1, G_1, G_2)

5. Среднее время в микросекундах за 10 запусков программы (прямое обращение/cv::pow): 2487/326

Коэффициент ускорения ~ 7.6

Текст программы

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

using namespace cv;
using namespace std;

int main() {

    //make empty template
    Mat img(180, 768, CV_8UC1);
    img = 0;
    Rect2d rc = { 0, 0, 768, 60 };
    rectangle(img, rc, { 100 }, 1);

    //create gray-scale 256b image
    Mat i_1 = img(rc);
```

```
for (int i = 0; i < 256; ++i) {
        for (int j = rc.y; j < rc.y + rc.height; ++j) {
            line(i_1, Point2d(i * 3, j), Point2d(i * 3 + 2, j), Scalar(i), 1);
        }
    }
    //direct gamma-correction method with gamma = 2.3
    Mat g_2 = i_1.clone();
    g_2.convertTo(g_2, CV_32F, 1. / 255);
    //check time for direct method
    chrono::steady_clock::time_point begin_t_direct =
chrono::steady_clock::now();
    for (int i = 0; i < g_2.cols; ++i) {
        for (int j = 0; j < g_2.rows; ++j) {
            g_2.at<float>(j, i) = pow(g_2.at<float>(j, i), 2.3);
        }
    }
    chrono::steady_clock::time_point end_t_direct = chrono::steady_clock::now();
    cout << chrono::duration_cast<chrono::microseconds>(end_t_direct -
begin_t_direct).count() << " micro_s" << endl;</pre>
    g_2.convertTo(g_2, CV_8UC1, 255);
    rc.y += rc.height;
    //copy to template
    g_2.copyTo(img(Rect(rc.x, rc.y, g_2.cols, g_2.rows)));
    //draw border to separate images
    rectangle(img, rc, { 250 }, 1);
    //cv::pow gamma-correction method with gamma = 2.3
    Mat g_3 = i_1.clone();
    g_3.convertTo(g_3, CV_32F, 1. / 255);
    //check time for cv::pow method
    chrono::steady_clock::time_point begin_t_pow = chrono::steady_clock::now();
    pow(g_3, 2.3, g_3);
    chrono::steady_clock::time_point end_t_pow = chrono::steady_clock::now();
    cout << chrono::duration_cast<chrono::microseconds>(end_t_pow -
begin_t_pow).count() << " micro_s" << endl;</pre>
    g_3.convertTo(g_3, CV_8UC1, 255);
    rc.y += rc.height;
    //copy to template
    g_3.copyTo(img(Rect(rc.x, rc.y, g_3.cols, g_3.rows)));
    //draw border to separate images
    rectangle(img, rc, { 150 }, 1);
    //show result
    imshow("laba_1", img);
    waitKey(0);
```

```
//save result
imwrite("lab01.png", img);
}
```