Работа 1. Исследование гамма-коррекции

автор: Мырзаханов М.А.
дата: 2022-02-16T18:00:00

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 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.

Результаты

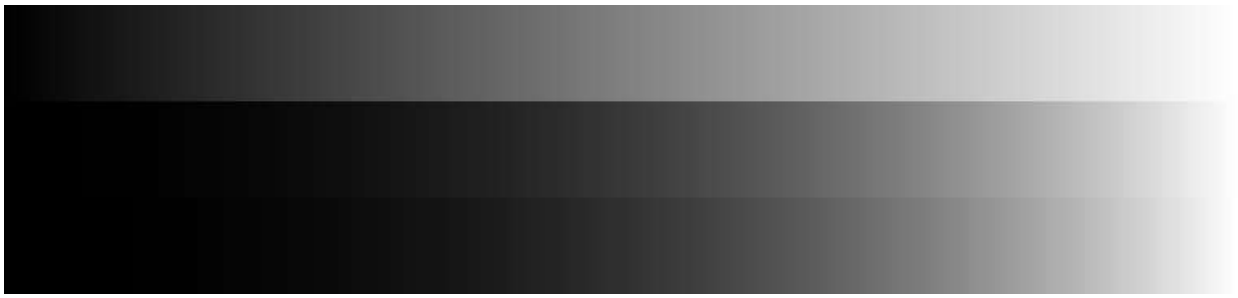

Рис. 1. Результаты работы программы (сверху вниз I_1, G_1, G_2)

Результаты замера времени обработки изображений:

п.2 - 934 microseconds
п.3 - 1287 microseconds

Текст программы

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

using namespace std::chrono;

int main() {
  cv::Mat img(180, 768, CV_8UC1);
  // draw dummy image
  img = 0;
  double alpha = 2.2;
  double beta = 2.4;
  cv::Rect2d rc = { 0, 0, 768, 60 };
  //1
  rc.y += rc.height;
```

```
  for (int y = 0; y < 180; y++)
    for (int x = 0; x < 768; x++)
      img.at<uchar>(y, x) = x / 3;
  //2
  auto start = high_resolution_clock::now();
  cv::Mat new_img(img);
  new_img.convertTo(new_img, CV_64FC1, 1.0f / 255.0f);
  cv::pow(new_img, alpha, new_img);
  new_img.convertTo(new_img, CV_64FC1, 255.0f);
  new_img(rc).copyTo(img(rc));
  auto finish = high_resolution_clock::now();
  auto elapsed = duration_cast<microseconds>(finish - start);

  std::cout << elapsed.count() << " microseconds" << std::endl;

  //3
  auto start2 = high_resolution_clock::now();
  rc.y += rc.height;
  cv::Mat new_img2(img);
  for (int y = 120; y < 180; y++)
    for (int x = 0; x < 768; x++)
      new_img2.at<cv::uint8_t>(y, x) = pow(img.at<cv::uint8_t>(y, x) / 255.0f, beta)
* 255.0f;
  new_img2(rc).copyTo(img(rc));
  auto finish2 = high_resolution_clock::now();
  auto elapsed2 = duration_cast<microseconds>(finish2 - start2);

  std::cout << elapsed2.count() << " microseconds" << std::endl;

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