МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и цифровых технологий

Кафедра КБ-14 «Цифровые технологии обработки данных»

Платформы анализа больших данных

Лабораторная работа 1

Вариант 6. Применить гамма-коррекцию к изображению

Выполнил:

Студент группы БСБО-09-22

Шутов Кирилл Сергеевич

Проверил:

Кашкин Евгений Владимирович

Москва, 2025

Постановка задачи

Реализовать на CUDA программу для применения гамма-коррекции к изображению с использованием stb_image для загрузки и stb_image_write для сохранения результата. Задача – обработать изображение на GPU, используя встроенные идентификаторы CUDA и обеспечить корректную работу при компиляции через nvcc (.cu файл).

Описание кода и выполненных действий

В начале работы происходит подключение библиотек stb_image_.h и stb_image_write_.h с определением макросов STB_IMAGE_IMPLEMENTATION и STB_IMAGE_WRITE_IMPLEMENTATION, что позволяет загрузить изображение с диска и сохранить результат после обработки.

Функция stbi_load загружает изображение «input.png», в массив байтов, преобразуя его в RGB-формат с тремя каналами, что упрощает дальнейшую обработку.

Затем выделяется память для исходного и результирующего изображений. С помощью cudaMalloc выделяется память на устройстве (GPU), и данные копируются туда через cudaMemcpy.

Далее реализуется CUDA-ядро gammaCorrectionKernel, которое обрабатывает каждый пиксель отдельно. Используя встроенные переменные blockIdx, threadIdx и blockDim, вычисляется координата пикселя. Затем производится нормализация значения канала (0–1), применяется функция гамма-коррекции с использованием powf(normalized, gamma), и значение денормализуется обратно (0–255).

Ядро запускается с использованием 2D-сетки и 2D-блоков для параллельной обработки. После завершения работы ядра выполняется синхронизация с помощью cudaDeviceSynchronize, а затем данные копируются обратно на хост.

Сохранённое обработанное изображение записывается в файл с помощью функции stbi_write_png, например, «output.png».

Готовую версию кода можно увидеть ниже (см. листинг 1).

```
#include <cuda_runtime.h>
#include <device_launch_parameters.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
#define STB_IMAGE_WRITE_IMPLEMENTATION
#include "stb_image_write.h"

__global__ void gammaCorrectionKernel(unsigned char* input, unsigned char* output,
int width, int height, float gamma) {
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;

    if (x < width && y < height) {
        int idx = (y * width + x) * 3;
        for (int i = 0; i < 3; i++) {
            float normalized = input[idx + i] / 255.0f;
            float corrected = powf(normalized, gamma);
            output[idx + i] = (unsigned char)(corrected * 255.0f);
        }
    }
}

int main() {
    int width, height, channels;
    unsigned char* h_input = stbi_load("input.png", &width, &height, &channels, 3);
    if (!h_input) {
        fprintf(stderr, "Не удалось загрузить изображение\n");
        return 1;
    }
    size_t imageSize = width * height * 3 * sizeof(unsigned char);
    unsigned char* h_output = (unsigned char*)malloc(imageSize);
    unsigned char* d_input, * d_output;
    cudaMalloc((void**)&d_input, imageSize);
    cudaMalloc((void**)&d_output, imageSize);
    cudaMemcpy(d_input, h_input, imageSize, cudaMemcpyHostToDevice);
    dim3 blockSize(16, 16);
    dim3 gridSize((width + blockSize.x - 1) / blockSize.x, (height + blockSize.y -
1) / blockSize.y);
    float gamma = 2.2f;
    gammaCorrectionKernel << <gridSize, blockSize >> > (d_input, d_output, width,
height, gamma);
    cudaDeviceSynchronize();
    cudaMemcpy(h_output, d_output, imageSize, cudaMemcpyDeviceToHost);
    stbi_write_png("output.png", width, height, 3, h_output, width * 3);
    cudaFree(d_input);
    cudaFree(d_output);
    stbi_image_free(h_input);
    free(h_output);
    return 0;
}
```

Листинг 1. Программа для гамма-коррекции

Программа обрабатывает изображения применяя гамма-коррекции.

Исходное изображение представлено на рисунке 1.

Рисунок 1. Исходное изображение

На рисунке 2 представлено итоговое изображение после гамма коррекции.

Рисунок 2. Итоговое изображение

Вывод

Была создана эффективная CUDA-программа, применяющая гамма-коррекцию к изображению. Использование библиотек stb_image и stb_image_write обеспечивает простоту работы с форматами файлов, а обработка на GPU позволяет масштабировать задачу на большие изображения.

Основные сложности были связаны с правильной конфигурацией компиляции CUDA-кода, что решается использованием nvcc и корректного расширения файла.

Источники

1. Документация NVIDIA CUDA. [Электронный ресурс] URL: https://docs.nvidia.com/cuda/ Дата обращения: (03.03.2025 г).
2. Репозиторий stb. [Электронный ресурс] URL: https://github.com/nothings/stb Дата обращения: (03.03.2025 г).
3. NVIDIA 2D Image And Signal Performance Primitives [Электронный ресурс] URL: https://docs.nvidia.com/cuda/archive/11.0/npp/group__image__color__gamma__correction.html Дата обращения: (03.03.2025 г).
4. Подробнее о разработке софта рентгеновского томографа [Электронный ресурс] URL: https://habr.com/ru/companies/edison/articles/282848/ Дата обращения: (03.03.2025 г).