|  |
| --- |
| Изображение выглядит как зарисовка, рисунок, символ, корона  Контент, сгенерированный ИИ, может содержать ошибки. |
| 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.

Изображение выглядит как снимок экрана, небо

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1. Исходное изображение

На рисунке 2 представлено итоговое изображение после гамма коррекции.

Изображение выглядит как снимок экрана, небо, темнота, ночь

Контент, сгенерированный ИИ, может содержать ошибки.

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).