

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

**Лабораторная работа №3  
по курсу «Основы CUDA»**

***Классификация и кластеризация изображений на GPU.***

Выполнил: *Ши Хуэй*  
Группа: 638  
Преподаватели: А.Ю. Морозов,  
Е.Е. Заяц

Москва, 2025

## **Условие**

**Цель работы.** Научиться использовать GPU для классификации и кластеризации изображений. Использование *константной памяти и одномерной сетки потоков*. Формат изображений соответствует формату описанному в лабораторной работе 2. Во всех вариантах, в результирующем изображении, на месте альфа-канала должен быть записан номер класса(кластера) к которому был отнесен соответствующий пиксель. Если пиксель можно отнести к нескольким классам, то выбирается класс с наименьшим номером.

Оценка вектора средних и ковариационной матрицы:

### **Вариант 2 - Метод расстояния Махalanобиса.**

Для некоторого пикселя , номер класса определяется следующим образом:

## **Программное и аппаратное обеспечение**

Работа выполнялась на вычислительном кластере «Полюс» Московского государственного университета имени М. В. Ломоносова.

Кластер состоит из 5 вычислительных узлов с пиковой производительностью **55.84 TFlop/s** и производительностью по тесту Linpack — **40.39 TFlop/s**.

### **На каждом вычислительном узле:**

- **Процессоры:**  $2 \times$  IBM Power 8
- **Пиковая производительность DP:**  $\approx 580$  GFlop/s ( $290 \times 2$ )
- **Графические процессоры:**  $2 \times$  NVIDIA Tesla P100
- **Число процессорных ядер:** 20 (по 10 на каждом CPU)
- **Число потоков на ядро:** 8
- **Оперативная память:** 256 ГБ (в отдельных узлах до 1024 ГБ)
- **Коммуникационная сеть:** Infiniband / 100 Gb
- **Система хранения данных:** GPFS
- **Операционная система:** Linux Red Hat 7.5

### **Характеристики GPU — NVIDIA Tesla P100:**

- **Compute Capability:** 6.0
- **Общая глобальная память:** 16 ГБ (16384 МиБ)
- **Shared memory per block:** 49152 байт
- **Constant memory:** 65536 байт
- **Registers per block:** 65536
- **Максимальное число нитей в блоке:** 1024 (1024, 1024, 64)
- **Максимальное число блоков:** (2 147 483 647, 65 535, 65 535)
- **Количество мультипроцессоров:** 56

### **Программное обеспечение:**

- **Операционная система:** Linux Red Hat 7.5 (на узлах Polus)
- **Среда разработки:** SSH-доступ к кластеру Polus через Visual Studio Code или nano
- **Компилятор:** nvcc 12.0 (с поддержкой g++ 13.3.0)
- **CUDA Toolkit:** версия 12.0
- **Дополнительные инструменты:** MPI, OpenMP (по необходимости), система очередей LSF для запуска задач
- **Библиотеки:** стандартные CUDA API, без использования внешних (Thrust, cuBLAS и др.)

## Метод решения

В работе реализован классификатор пикселей цветного изображения в пространстве признаков RGB на основе расстояния Махalanобиса. В качестве обучающей информации используются координаты опорных пикселей для каждого класса: по ним на стороне CPU оцениваются параметры распределения цветов класса, а на стороне GPU выполняется массовая классификация всех пикселей исходного изображения. Сначала программа считывает изображение и наборы координат образцов для каждого класса. По значениями каналов R, G и B, извлечённым в этих координатах, для каждого класса вычисляется вектор средних и ковариационная матрица в несмешённой постановке. Для предотвращения вырождения при малом числе образцов или сильной корреляции признаков матрица дополнительно стабилизируется небольшой добавкой по диагонали; после этого она обращается на CPU. Полученные векторы средних и обратные ковариационные матрицы упаковываются в компактные массивы.

Далее параметры всех классов копируются в константную память устройства. Это позволяет всем потокам обращаться к одним и тем же неизменяемым данным с минимальными задержками и уменьшает нагрузку на глобальную память. Само изображение передаётся в глобальную память GPU в виде линейного буфера пикселей. Классификация выполняется одним ядром CUDA с одномерной сеткой потоков, где каждая нить отвечает ровно за один пиксель. Поток считывает три цветовых компонента пикселя, последовательно перебирает все классы, для каждого класса вычисляет соответствующую квадратичную форму, эквивалентную расстоянию Махalanобиса, и находит минимальное значение. При равенстве критериев выбирается класс с меньшим номером, что однозначно определяет результат для каждого пикселя. По завершении вычислений поток записывает в альфа-канал выходного пикселя номер выбранного класса; цветовые каналы при этом сохраняются без изменений (или могут быть переопределены на этапе визуализации, если это требуется экспериментом).

Такое разбиение на этапы обеспечивает хорошее совмещение вычислений и обменов: трудоёмкая по количеству пикселей часть переносится на GPU, а небольшая по объёму статистическая предобработка выполняется на CPU. Использование константной памяти для параметров классов сокращает число обращений к глобальной памяти, а одномерная конфигурация упрощает адресацию и обеспечивает коалесцированные чтения и записи. Итоговая вычислительная сложность линейна по числу классов на пиксель, хорошо масштабируется с ростом размера изображения и демонстрирует

стабильную производительность за счёт фиксированной размерности признакового пространства.

## Описание программы

Программа состоит из следующих основных компонентов:

1. Чтение данных

Из файла `in.data`читываются размеры изображения (`w, h`) и пиксельные данные (`r, g, b, a`).

2. Передача данных на GPU

Память для входных и выходных массивов выделяется через `cudaMalloc`.

В константную память копируются массивы средних и ковариационных матриц (`cudaMemcpyToSymbol`).

3. Основное ядро — `classify_pixels<<<gridSize, blockSize>>>`

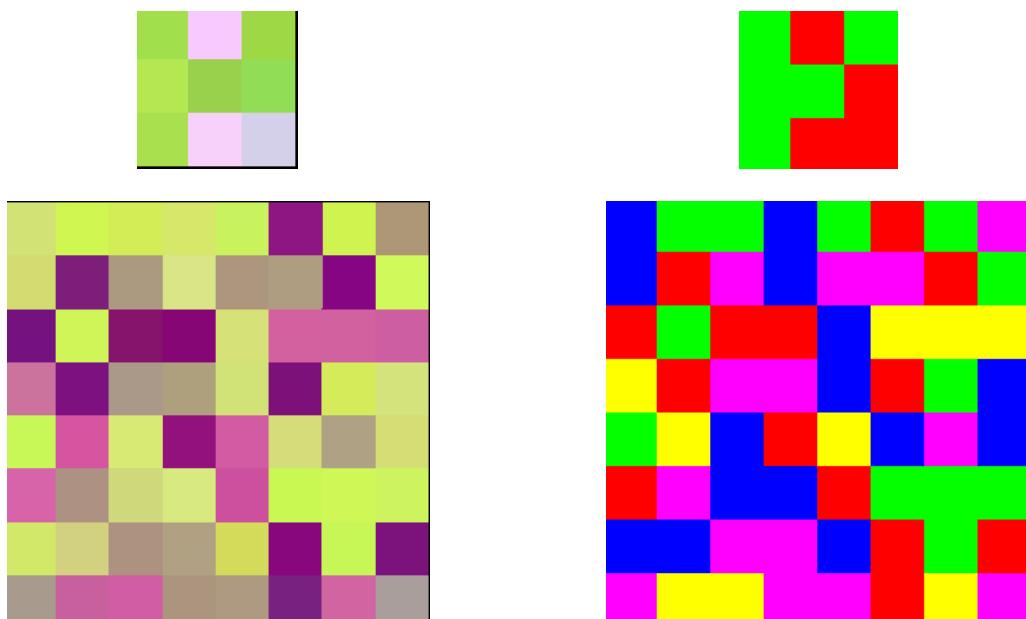
4. Запись результата

После копирования массивов обратно на CPU (`cudaMemcpy`) результаты сохраняются в файл `out.data`, аналогично формату ЛР 2.

5. Освобождение ресурсов

Все буферы и указатели GPU освобождаются через `cudaFree`.

## Результаты



## Выводы

Реализован классификатор пикселей на **расстоянии Махalanобиса** (вариант 2)