УДК 519.688:004.272

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ РАСЧЁТА РАССТОЯНИЯ ХЭММИНГА С ИСПОЛЬЗОВАНИЕМ CUDA ДЛЯ СИСТЕМ БИОМЕТРИЧЕСКОЙ ИДЕНТИФИКАЦИИ ПО РАДУЖКЕ ГЛАЗА

ШАЙДУЛЛИН Марсель Рамилевич
Научный руководитель: **ШАКИРЗЯНОВ Ринат Михайлович**
кандидат технических наук, доцент кафедры прикладной математики и информатики
Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ
г. Казань, Россия

Биометрическая идентификация по радужной оболочке глаза основана на сравнении двоичных шаблонов, извлечённых из текстуры радужки. При верификации в режиме «один ко многим» вычисление нормированного расстояния Хэмминга между пробным шаблоном и каждой записью базы данных имеет вычислительную сложность O(N × L), где N – число субъектов в галерее, L – длина бинарного кода.. Для шаблона длиной L = 8192 последовательная реализация на CPU при масштабировании N становится узким местом интерактивных систем. В работе представлено программное обеспечение, реализующее полный конвейер извлечения iris-кода и параллельное вычисление расстояния Хэмминга на GPU с использованием библиотеки CuPy (CUDA 12). Пакетный режим сравнения одного шаблона с матрицей галереи N × L выполняется посредством поэлементного XOR и параллельной редукции по строкам на устройстве. На конфигурации AMD Ryzen 7 5700X (до 4,32 ГГц), 32 ГБ DDR4-3200, NVIDIA GeForce RTX 3060 12 ГБ при N = 2000 достигнуто ускорение пакетного поиска ×93,8 (CPU: 51,40 мс; GPU: 0,55 мс). Для попарного сравнения двух шаблонов GPU медленнее CPU примерно в 21 раз (0,1044 мс против 0,0050 мс) из-за доминирования накладных расходов передачи данных и запуска ядра при малом объёме вычислений.

Ключевые слова: биометрия, радужка глаза, расстояние Хэмминга, CUDA, CuPy, параллельные вычисления, идентификация личности, HPC.

Цель работы – разработка и экспериментальная оценка реализации расчёта нормированного расстояния Хэмминга на графическом процессоре в составе программного комплекса биометрической идентификации по радужке. Задачи: реализовать конвейер формирования двоичного шаблона и сравнения на центральном и графическом процессорах; обеспечить пакетный поиск по базе шаблонов; провести сравнительный замер времени для попарного и пакетного режимов; определить область применимости параллельных вычислений.

Проблематика.

Биометрические системы по радужке относятся к точным неконтактным методам аутентификации благодаря устойчивости текстурных признаков и компактному двоичному представлению [5]. На этапе идентификации «1:N» центральной операцией является расстояние Хэмминга – доля несовпадающих бит между шаблонами [6]. Пусть p – пробный шаблон длины L, G – матрица галереи N×L; для каждой строки g_i вычисляется HD(p, g_i) = popcount(p ⊕ g_i) / L, где ⊕ – исключающее «или». При L = 8192 и N порядка 10^3–10^6 суммарная трудоёмкость O(N·L) ограничивает пропускную способность интерактивных систем. Графические процессоры обеспечивают массовый параллелизм и высокую пропускную способность памяти, что делает целесообразным перенос пакетного сравнения на архитектуру CUDA [1; 3]; при этом единичное попарное сравнение на CPU выполняется за микросекунды и требует отдельной оценки.

Основное содержание.

В разработанном программном обеспечении изображение радужки обрабатывается на CPU в четыре этапа [4; 6]: (1) локализация зрачка и внешней границы радужки (градиентный поиск, преобразование Хафа, CLAHE, подавление

зеркальных бликов); (2) нормализация «резинового листа» – развёртка кольца в матрицу 32×256; (3) фильтрация банком фильтров Габора (четыре ориентации); (4) бинаризация по знаку отклонения от медианы. Итоговый шаблон – вектор L = 8192 бит; порог совпадения HD < 0,35.

На CPU для двух шаблонов a и b: HD = popcount(a ⊕ b) / L; применяется NumPy [7]. Для пакетного поиска реализованы векторизованный CPU (XOR матрицы с пробным вектором, редукция по строкам) и эталонный последовательный перебор по N записям (используется в бенчмарке). Клиентская часть (веб-интерфейс) отправляет изображения на REST API; сервер извлекает коды, вычисляет HD и замеряет время CPU и GPU. База шаблонов хранится в JSON; для демонстрации масштабирования генерируется N синтетических записей. Форматы данных радужки регламентируются ГОСТ Р 58295-2018 (ГОСТ Р 58295-2018 (ИСО/МЭК 19794-6:2011). Информационные технологии. Биометрия. Форматы обмена биометрическими данными. Часть 6. Данные изображения радужной оболочки глаза. – Москва : Стандартинформ, 2019. – 44 с.).

Механизмы реализации.

Пользовательское CUDA-ядро не программировалось; параллелизация выполнена через CuPy – NumPy-совместимый интерфейс к CUDA 12 (https://docs.cupy.dev/en/stable/; https://pypi.org/project/cupy-cuda12x/). На Windows перед импортом CuPy регистрируются пути к DLL NVRTC. При попарном сравнении на GPU вычисляется XOR двух векторов длины L и подсчитывается число несовпадающих бит. При пакетном поиске пробный вектор P и матрица G (N×L) размещаются на устройстве; формируется X = G ⊕ P (broadcasting по строкам); для каждой строки HD_i = popcount(X_i) / L. Распараллеливается XOR над N·L битами и редукция по N строкам [4; https://docs.nvidia.com/cuda/cuda-c-programming-guide/]. При бенчмарке: конвертация в uint8, однократная загрузка на GPU, прогрев, цикл вычислений на устройстве, synchronize() для замера (https://docs.nvidia.com/cuda/cuda-c-programming-guide/). При недоступности CUDA – автоматический переход на CPU.

Достигнутые результаты и выводы.

Эксперименты: CPU AMD Ryzen 7 5700X (до 4,32 ГГц), GPU NVIDIA GeForce RTX 3060 12 ГБ, ОЗУ 32 ГБ DDR4-3200, ОС Windows; стек Python, FastAPI, NumPy, CuPy (CUDA 12); L = 8192 бит; 50 итераций (пара), 15 (batch). Замеры – через встроенный бенчмарк веб-интерфейса.

Таблица 1

РЕЗУЛЬТАТЫ ПОПАРНОГО СРАВНЕНИЯ ДВУХ ШАБЛОНОВ

Реализация	Среднее время, мс	Примечание
CPU (NumPy)	0,0050	эталон
GPU (CuPy/CUDA)	0,1044	в ~21 раз медленнее CPU
Отношение T_GPU/T_CPU	-	≈ 20,9

При L = 8192 на GPU доминируют накладные расходы запуска ядра и синхронизации; перенос попарного HD на CUDA нецелесообразен (закон Амдала [10]).

Таблица 2

РЕЗУЛЬТАТЫ ПАКЕТНОГО ПОИСКА В БАЗЕ ШАБЛОНОВ

N	CPU, мс	GPU, мс	Ускорение S = T_CPU/T_GPU
200	4,6430	2,4157	×1,92
2000	51,3988	0,5479	×93,8

При N = 200 объём $N \cdot L \approx 1{,}64 \cdot 10^6$ недостаточен для полной компенсации затрат CUDA. При N = 2000 объём $\approx 1{,}64 \cdot 10^7$; время CPU выросло ~в 11 раз, время GPU снизилось за счёт амортизации фиксированных затрат. Ускорение ×93,8 подтверждает целесообразность GPU для идентификации «один ко многим» при промышленном масштабе базы.

Ограничения: извлечение шаблона на CPU может доминировать в полном времени ответа [11]; в бенчмарке – синтетическая база; на CPU применён намеренно последовательный цикл; отсутствие собственного CUDA-ядра ограничивает оптимизацию (упаковка бит, shared memory).

Выводы: разработано ПО с REST-интерфейсом, формирующее шаблоны 8192 бита и вычисляющее HD на CPU и GPU; пакетное сравнение перенесено на CUDA с однократной загрузкой данных. Реализация эффективна для «один ко многим» при N от нескольких сотен и выше и не предназначена для единичного попарного сравнения. Перспективы: собственное CUDA-ядро, перенос фильтрации Габора на GPU, интеграция с реальной галереей.

ЛИТЕРАТУРА

1. *Боресков, А. С.* Параллельные вычисления на GPU : архитектура и программная модель CUDA : учебное пособие / А. С. Боресков, Д. А. Харламов, Н. Г. Марковский. – Москва : ДМК Пресс, 2012. – 352 с.

2. *Воеводин, В. В.* Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – 2-е изд., испр. и доп. – Санкт-Петербург : Лань, 2020. – 664 с.

3. *Сандерс, Дж.* Технология CUDA в примерах : введение в программирование графических процессоров / Дж. Сандерс, Э. Кэндрот ; пер. с англ. – Москва : ДМК Пресс, 2018. – 304 с.

4. *Bradski G., Kaehler A.* Learning OpenCV : computer vision with the OpenCV library / G. Bradski, A. Kaehler. – Sebastopol : O'Reilly Media, 2008. – 575 p..

5. *Daugman, J.* High confidence visual recognition of persons by a test of statistical independence / J. Daugman // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1993. – Vol. 15, No. 11. – P. 1148-1161.

6. *Daugman, J.* How iris recognition works / J. Daugman // IEEE Transactions on Circuits and Systems for Video Technology. – 2004. – Vol. 14, No. 1. – P. 21-30.

7. *Harris, C. R.* Array programming with NumPy / C. R. Harris, K. J. Millman, S. J. van der Walt et al. // Nature. – 2020. – Vol. 585. – P. 357-362. DOI: 10.1038/s41586-020-2649-2.

IMPROVING HAMMING DISTANCE COMPUTATION EFFICIENCY USING CUDA FOR IRIS BIOMETRIC IDENTIFICATION SYSTEMS

SHAKIRZYANOV Marcel Ramilovich
Scientific Supervisor: **SHAKIRZYANOV Rinat Mikhailovich**
Candidate of Sciences in Technology, Associate Professor
Kazan National Research Technical Institute named after A. N. Tupolev – KAI
Kazan, Russia

Iris biometric identification is based on comparing binary templates extracted from iris texture. In one-to-many verification, computing normalized Hamming distance between a probe template and each database record has computational complexity O(N × L), where N is the gallery size and L is the binary code length. For templates of length L = 8192, sequential CPU implementation becomes a bottleneck for interactive systems as N scales. This paper presents software implementing a full iris-code extraction pipeline and parallel Hamming distance computation on GPU using CuPy (CUDA 12). Batch comparison of one template against an N × L gallery matrix is performed via element-wise XOR and parallel row-wise reduction on the device. On AMD Ryzen 7 5700X (up to 4.32 GHz), 32 GB DDR4-3200, NVIDIA GeForce RTX 3060 12 GB, at N = 2000, batch search speedup of ×93.8 *was achieved (CPU: 51.40 ms; GPU: 0.55 ms). For pairwise comparison of two templates, GPU is* slower *than CPU by approximately* 21 times *(0.1044 ms vs 0.0050 ms) due to dominant data transfer and kernel launch overhead at small computational volume.*

Keywords: biometrics, iris, Hamming distance, CUDA, CuPy, parallel computing, personal identification, HPC.