Использование графических процессоров мобильных устройств для решения задачи распознавания речи и биометрических признаков человека

Applying mobile graphics processing unit for speech and behavioral biometrics recognition

КОРОЛЕВ / KOROLEV A.
Алексей Игоревич
(a.nezachem@gmail.com)
аспирант,
СПбГУ, Факультет Прикладной математики – процессов управления,
Санкт-Петербург

ФИРУН / FIRUN K.
Константин Борисович
(Constantine@firun.ru, kfirun@cc.spbu.ru)
аспирант,
СПбГУ, Математическо-Механический факультет,
кафедра Системного программирования,
Санкт-Петербург

Ключевые слова: распределенные вычисления – speech recognition; распознавание речи – voice recognition; распознавание голоса – grid computing; графические процессоры – GPGPU.

Графические процессоры со сверхмалым энергопотреблением – например, в смартфонах – становятся неотъемлемой частью мобильных устройств, поэтому стали актуальны исследования возможностей применения данных процессоров в решении задач общего назначения, аналогично применению графических процессоров в серверных и персональных компьютерах. Наши исследования сфокусированы на применении графических процессоров со сверхмалым энергопотреблением для оптимизации приложений реального времени, работающих на мобильных устройствах с батарейным питанием. В данной работе мы рассматриваем использование графических процессоров мобильных устройств на примере задач по распознаванию речи и биометрических признаков человека – лица и голоса. Реализованное нами решение на мобильном устройстве показывает, что использование графического процессора в роли сопроцессора позволяет значительно повысить производительность, и в то же время существенно уменьшить энергопотребление по сравнению с решением использующим только основной процессор мобильного устройства.

Mobile graphics processing units (GPU) with low-energy consumption are becoming integral part of mobile devices. Due to widespread of mobile devices research on using mobile GPU for general purpose computations is becoming actual. Research described in our paper is dedicated to use of mobile GPU for computing and time optimization of machine learning algorithms — face and speaker recognition. Our implemented solution proofs that use of mobile GPU considerably lowers energy consumption with simultaneous performance increase comparing to solutions using only central processing unit of conventional mobile devices.

ВВЕДЕНИЕ

За последние несколько лет стали актуальными исследования по использованию графических процессоров (GPGPU) в роли сопроцессоров общего назначения для оптимизации ресурсоёмких приложений. В научных публикациях отмечены примеры оптимизации ресурсоёмких приложений, чья производительность возросла от нескольких до сотен раз в зависимости от использования алгоритмов параллелизма и вычислительной мощности графических процессоров. На данный момент графические процессоры доступны на мобильных устройствах – смартфонах, а также портативных игровых приставках. Так как данные мобильные устройства работают от батарейного питания, то используемые в них графические процессоры обладают сверхмалым

энергопотреблением с предельным уровнем энергопотребления менее 1 Вт [1]. В результате графические процессоры мобильных устройств наделены меньшим количеством вычислительных ядер, меньшей пропускной способностью памяти и архитектурой отличающейся от персональных и серверных аналогов.

Так как большинство алгоритмов по оптимизации работы центрального и графического процессоров (CPU-GPU) разработаны для персональных и серверных платформ, то для оптимального использования вычислительных ресурсов мобильного устройства необходимо определить вычислительную мощность связки центрального и графического процессоров, а также переработать способы использования графического процессора в роли сопроцессора. Кроме того, для оптимизации энергопотребления мобильного приложения необходимо определить энергоэффективность связки центрального и графического процессоров мобильного устройства. В научной литературе есть найти несколько статей посвящённых использованию графического процессора мобильного устройства в качестве сопроцессора общего назначения. В статье [5] авторы разработали алгоритм фильтрации изображения использующий ядро размерностью 3 на базе графического процессора PowerVR на однокристальной системе TI OMAP3 [1], повысив производительность с 13 до 29 кадров в секунду через уменьшение точности переменных с плавающей запятой. В статье [4] авторы утверждают, что графический процессор PowerVR с тактовой частотой 111 МГц по сравнению с процессором ARM Cortex-A8 с тактовой частотой 555 МГц показал большую производительностью (0,41 против 0,11 кадров в секунду) и меньшее энергопотребление (232 мВт против 361 мВт) на низкоуровневых задачах по обработке изображений (Гауссово размытие, изменение цвета и размера изображения). Исследования в данных статьях производились на эмулированной версии фреймворка OpenCL [9], который пока что не доступен на большинстве мобильных устройств и макетных платах.

В данной статье рассматривается вычислительная мощность и энергоэффективность текущего поколения центрального и графического процессоров мобильных устройств на примере по распознаванию речи и биометрических признаков человека – лица и голоса. В статье сделаны обзор архитектуры графического процессора мобильного устройства и его программный интерфейс, сравнение графического процессора мобильного устройства по вычислительной мощности и энергоэффективности с центральным процессором мобильного устройства и графическим процессором персонального персонального компьютера. В статье даётся описание разработанного авторами приложения по распознаванию речи и биометрических признаков человека, демонстрирующее использование графического процессора мобильного устройства для повышения вычислительной производительности и энергоэффективности. В заключении статьи сделаны выводы исследования и описаны последующие шаги исследовательской работы.

ОБЗОР АРХИТЕКТУРЫ ГРАФИЧЕСКОГО ПРОЦЕССОРА МОБИЛЬНОГО УСТРОЙСТВА

Графический процессор мобильного устройства обычно встроен в прикладной процессор однокристальной системы, который также состоит из одного или нескольких центральных процессоров, цифровых сигнальных процессоров и других специализированных прикладных элементов, как показано на Рис. 1 вместо собственной видеопамяти, встроенный графический процессор находится на системной шине с остальными вычислительными элементами, имеющими доступ к внешней памяти, тем самым графический процессор имеет меньшую пропускную способность памяти, чем графические процессоры для персональных и серверных компьютеров [6]. Наиболее распространённые однокристальные системы со встроенным графическим процессором представлены следующими решениями: Qualcomm SnapDragon с графическим процессором Adreno [2], TI OMAP3 с графическим процессором PowerVR SGX [1] и Nvidia Tegra с графическим процессором GeForce со сверхмалым энергопотреблением [3].

Графические процессоры мобильных устройств обычно разработаны с акцентом на меньшее энергопотребление, чем на высокую производительность. Одной из основных методик по уменьшению энергопотребления мобильных устройств является уменьшение потока данных между графическим процессором и памятью [6]. Например, в графическом процессоре GeForce в системе Tegra для оптимизации обращений к памяти при отрисовке изображения реализована внутрикристальная кэш-память для хранения пикселей, текстур, вершин и других аттрибутов (см. Рис. 2). Встроенная кэш-память повышает вычислительную производительность, так как кэш-память обладает меньшей латентностью чем память вне кристалла [3]. Хранение текстур и буфера кадра целиком в кэш-памяти значительно уменьшает количество обращений к памяти вне кристалла. Для достижения такого результата прибегают к различным оптимизациям, например, графиче-

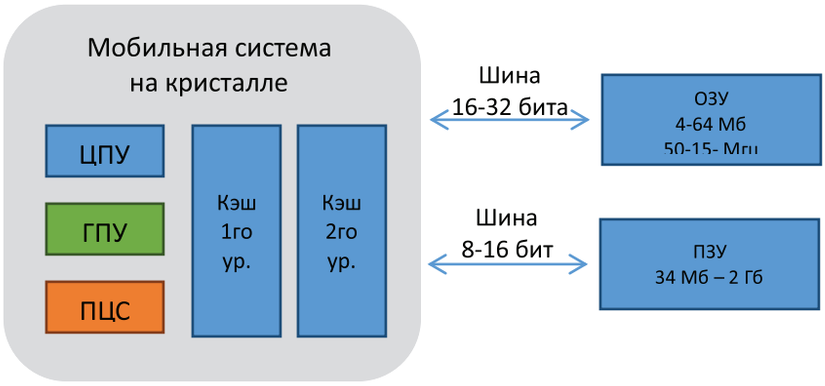

Рис. 1. Блок-схема однокристальной системы

ский процессор PowerVR использует алгоритм фрагментации кадра (обычно размер фрагмента не превышает 16x16 пикселей), в ходе которого в отрисовывается один фрагмент за раз, таким образом вся необходимая информация умещается во внутрикристальной кэш-памяти [6].

Другой способ для уменьшения количества обращений к памяти вне кристалла заключается в хранении сжатых данных (т.е. сжатых текстур, вершин и буферов кадра) в памяти вне кристалла, которые динамически распаковываются графическим процессором перед последующей обработкой [7]. Уменьшение количества обращений к памяти вне кристалла также помогает уменьшить энергопотребление. При отрисовке трёхмерной графики множество полигонов и соответствующие им пиксели могут накладываться на одну и ту же двухмерную область кадра. Традиционно графические процессоры отрисовывают все полигоны, включая скрытые глазу, и затем уменьшают число полигонов, оставляя только те, чья глубина отвечает требованиям кадра. Графический процессор в Tegra уменьшает число полигонов перед отрисовкой всех объектов, тем самым эффективно уменьшает количество лишних вычислений, как и ненужных обращений к памяти вне кристалла.

ПРОГРАММНЫЙ ИНТЕРФЕЙС

Основным программным интерфейсом для работы с графическими процессорами на мобильных устройствах является среда OpenGL ES [8], поддерживающий программируемый шейдер вершин и шейдер фрагментов для повершинной и попиксельной отрисовки соответственно. Остальные этапы отрисовки, включая растеризацию и уменьшение количества полигонов, остаются вшитыми в графический процессор функциями (fixed functions). OpenGL ES является подмножеством широко распространённого графического программного интерфейса OpenGL, который стал стандартом де-факто для персональных компьютеров, а также игровых консолей. OpenGL ES уменьшил избыточность программного интерфейса OpenGL – например, если несколько методов выполняют одну и ту же операцию, то в OpenGL ES оставлен лишь наиболее практичный метод, а избыточные удалены. Примером может служить то что в OpenGL ES только вершинные массивы определяют геометрию, в то время как в OpenGL возможно использование нескольких функций со схожим результатом. Также в OpenGL ES внедрены новые по сравнению с OpenGL функции, позволяющие обходить ограничения энергоэффективности, встречающиеся на мобильных устройствах. Например, для уменьшения энергопотребления и повышения производительности шейдеров были представлены спецификаторы точности: lowp (10-битный формат с фиксированной запятой в полуотрезке [-2, 2) с точностью 1/256), medium (16-битный формат с плавающей запятой на отрезке [-65520, 65520]) и highp (32-битный формат с плавающей запятой)[11].

Для использования графического процессора мобильного устройства в качестве сопроцессора, разработчикам приходится приводить алгоритмы к виду графических операций, создавая шейдерные программы, которые конфигурируют шейдеры вершин и фрагментов. Однако, графический программный интерфейс предоставляет мало возможностей по управлению низкоуровневым аппаратным обеспечением, тем самым усложняя использование графического процессора для вычислений общего назначения. К примеру, графический интерфейс OpenGL ES не располагает функцией “scatter” (рассыпания, т.е. записи данных в произвольный раздел памяти вне кристалла) или синхронизацией потоков. Часто используемые в персональных компьютерах интерфейсы

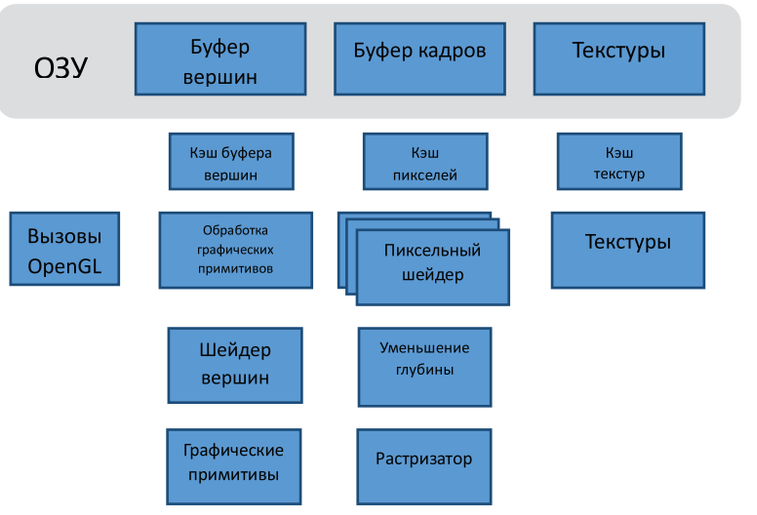

Рис. 2. Блок-схема работы графического процессора мобильного устройства на примере Qualcomm Snapdragon

высокого уровня, такие как CUDA [10] и OpenCL [9], пока что не поддерживаются в мобильных устройствах и встраиваемых платформах.

ГРАФИЧЕСКИЙ ПРОЦЕССОР МОБИЛЬНЫХ УСТРОЙСТВ КАК СОПРОЦЕССОР ОБЩЕГО НАЗНАЧЕНИЯ: ВОЗМОЖНОСТИ И ОГРАНИЧЕНИЯ

В данной статье приводится реализация быстрого преобразования Фурье (БПФ) – ключевого вычислительного метода множества алгоритмов распознавания образов и обработки сигналов, работающего на графическом процессоре мобильного устройства. Портирование БПФ на графический процессор позволяет оценить возможность применения графического процессора для решения более требовательных к вычислительным ресурсам задач, а также сравнить производительность и энергопотребление по сравнению с центральным процессором мобильного устройства и графическим процессором персонального компьютера. Экспериментальным стендом для исследований служила система на кристалле Qualcomm SnapDragon под управлением ОС Android 4.0.4 со следующей спецификацией: двух-ядерный центральный процессор ARM с тактовой частотой 1 ГГц, 1 Гб ОЗУ, графический процессор Adreno с тактовой частотой 333 МГц и 512 Мб ОЗУ.

РЕАЛИЗАЦИЯ БПФ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ.

Одномерное БПФ для N чисел определяется как

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-\frac{2\pi n k}{N}},$$

где $0 \le k \le N-1$. В данной статье рассматривается реализация алгоритма Кули-Тьюки, основанного на подходе описанном в [15] с использованием OpenGL ES и языка программирования шейдеров. Ход работы алгоритма Кули-Тьюки описан на Рис. 3. Для вычисления алгоритма для N чисел требуется $log_2 N$ шагов. Числа на каждом шаге образуют пары между двумя группами, как показано в пунктирном прямоугольнике на Рис. 3. Вычисление этой группы числе выражается формулой: $c=a+w^0*b$ и $c=a-w^0*b$. Коэффициенты каждого числа (т.е. $\pm w^0$) заранее вычислены и хранятся в виде текстуры для последующей обработки шейдерной программой. Следует заметить, что знак коэффициента также включается в текстуру для того чтобы избежать условного вычисления (т.е. ветвления) в шейдерной программе.

Трансформация двумерного изображения выполняется через последовательное применение одномерного БПФ к столбцам и колонкам. На каждом шаге вычисляется четыре числа покрывающих целый двумерный массив мощностью NxN. На каждом шаге требуется текстура размером $Nx1$ для хранения заранее вычисленных коэффициентов. Для экономии пропускной способности памяти между памятью вне кристалла и кэш-памятью графического процессора коэффициенты для реальной и мнимой частей сохраняются в двух каналах текстуры. Также необходима и вторая текстура, в которой хранятся полученные индексы. Хотя можно объединить эти две текстуры, но такой подход нерационален, так как индексам требуется меньшая по размеру текстура, чем для коэффициентов. Таким образом, выделяется текстура меньшего размера для хранения полученных индексов. Итерационный процесс

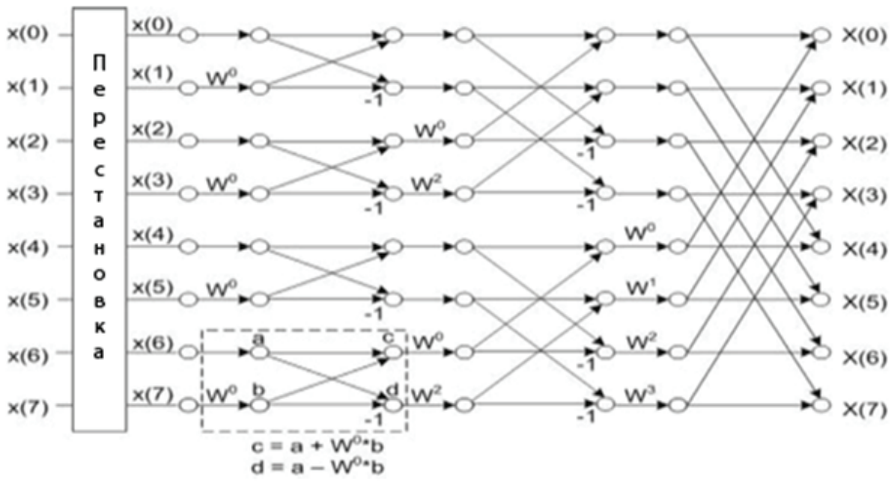

Рис. 3. Поточная обработка 8-значного одномерного БПФ

вычислений достигается благодаря нескольким проходам отрисовки и буфером кадра для чисел с плавающей запятой, в котором хранятся промежуточные результаты отрисовки.

СРАВНЕНИЕ С ЦЕНТРАЛЬНЫМ ПРОЦЕССОРОМ МОБИЛЬНОГО УСТРОЙСТВА

Измеренные время исполнения и энергопотребления двумерного БПФ различных размеров на экспериментальном стенде указаны в первой и второй строках Таблицы 1 соответственно. Указанное время исполнения является средним временем вычисления БПФ и ОБПФ, проходившим 50 раз подряд. Для БПФ на двумерном массиве 128x128 графический процессор оказался в 3 раза быстрее и на 8% менее энергоэффективным, чем центральный процессор мобильного устройства (1 секунда против 3,1 секунд, 4,1 Вт против 3,7 Вт). Чуть большее энергопотребление вызвано тем, что при использовании графического процессора центральный процессор всё равно не простаивает и потребляет энергию. В результате соотношение всей потреблённой энергии центрального процессора и графического процессора составляет 2,86 против 1. Увеличение быстродействия графического процессора по сравнению с центральным процессором падает до 2,2 и 1,3 раз при вычислении БПФ массивов мощностью 256x256 и 1024x1024 соответственно. Для БПФ массивов большей мощности потребуется использование памяти вне кристалла, так как кэш-память графического процессора будет недостаточна для хранения данных. Улучшение производительности центрального процессора при вычислении БПФ массивов большой мощности происходит, потому что объём кэш-памяти у центрального процессора больше чем у графического процессора. Кроме того, графический процессор простаивает во время ожидания данных из памяти вне кристалла. Таким образом энергопотребление графического процессора при вычислении БПФ массивов большой мощности чуть ниже чем в случае с массивами меньшей мощности (3,6 Вт против 4,1 Вт).

СРАВНЕНИЕ С ГРАФИЧЕСКИМ ПРОЦЕССОРОМ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Также были измерены результаты аналогичных вычислений на персональном компьютере, оснащённом графическим процессором Nvidia GeForce 8800. Полученные результаты можно увидеть на третьей строке в Таблице 1. Энергопотребление указанное в таблице обозначает суммарное потребление всей вычислительной системы – экспериментального стенда и персонального компьютера соответственно. Рассеяние мощности и время вычисления БПФ для графического процессора персонального компьютера указанное в таблице 1 были представлены в работах [16] и [17] соответственно.

Несмотря на большее время требуемое графическим процессором мобильного устройства для вычисления БПФ по сравнению с GeForce 8800, мобильное устройство потребляет значительно меньше энергии по сравнению с персональным компьютером. При вычислении БПФ массивов малой мощности (например, 128x128) графический процессор мобильного устройства достигает

Таблица 1.

Сравнение быстродействия и энергоэффективности центрального и графического процессоров мобильного устройства с персональным компьютером

		128x128	256x256	1024x1024
Центральный процессор Snapdragon (ARM Cortex A9)	Врем (мс)	31	124	1953
	Мощность (Вт)	3,67	3,69	3,57
	Энергия (мДж)	113,7	457,5	6972,2
Графический процессор Adreno	Время	10	55	1450
	Мощность (Вт)	4,10	3,70	3,65
	Энергия (мДж)	41	203,5	5292,5
GeForce 8800 + Intel i7	Время (мс)	0,11	0,17	1,74
	Мощность (Вт)	483,89	483,89	483,89
	Энергия (мДж)	53,2	82,11	841,96

лучшей эффективности использования энергии (41 мДж против 53 мДж). При вычислении БПФ массивов большей мощности, данное преимущество исчезает из-за ограниченного объёма кэш-памяти графического процессора мобильного устройства.

Графический процессор зависит от центрального процессора, который создаёт данные вершин и текстур, которые в последующем посылаются в кэш-память графического процессора, и тем самым запускают вычисление на нём. Для вычисления БПФ на запуск графического процессора требуется порядка 0,4 секунды. Для получения потенциального ускорения вычисления, которое может предоставить графический процессор, требуется чтобы процесс выполняющейся на графическом процессоре был вычислительно-интенсивным, тем самым компенсирующим потерю на время запуска графического процессора.

Кроме того, для вычисления БПФ, а также для многих других итеративных вычислений с плавающей запятой, требуются текстуры и буферы кадра с плавающей запятой. В программном интерфейсе OpenGL ES текстуры с плавающей запятой поддерживаются, если включена поддержка дополнения *OES_texture_float*. Хотя поддержка *OES_texture_float* не обязательно значит, что приложение поддерживает буфер кадра с плавающей запятой – под такой буфер может быть выделены ресурсы, но драйвер графического процессора может не дать к нему доступ разработчикам приложений. Среди трёх наиболее распространённых графических процессоров – Nvidia Tegra [3], PowerVR SGX [1] и Adreno [2], только Tegra поддерживает буфер кадра с плавающей точкой. Без буфера кадра с плавающей точкой разработчики вынуждены либо использовать вычисление с фиксированной запятой и жертвовать точностью, либо преобразовывать значение с плавающей запятой в вид RGBA8888 для хранения в буфере, с последующим извлечением из буфера и обратным преобразованием в формат с плавающей точкой (таким образом значение соответствует формату вывода изображения на экран). Подобные преобразования вызывают значительные временные задержки и могут полностью свести на нет преимущества использования графического процессора для вычислений общего назначения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ПО РАСПОЗНАВАНИЮ РЕЧИ, ГОЛОСА И ЛИЦА ЧЕЛОВЕКА

На Рис. 4 показано мобильное приложение с двухфакторной биометрической идентификацией (по лицу и голосу) с поддержкой голосовых команд для управления: приложение проводит авторизацию пользователя по его лицу и голосовому паролю, после чего пользователь получает возможность производить системные настройки своего

мобильного устройства при помощи голосовых команд. Распознавание лица и голоса человека являются основными технологиями используемыми в описываемом приложении. Процесс их работы выглядит следующим образом: 1) Автоматическое определение лица в кадре, в ходе которого обрабатывается всё изображение для поиска областей с лицом человека; 2) Локализация опознавательных точек лица – определение областей изображения с глазами, носом и ртом. В последующем локализованная область подгоняется под требуемый размер; 3) Выделение признаков лица в локализованной области изображения – то есть определяются признаки лица инвариантные и устойчивые к ошибкам при различных условиях освещённости, положения и выражения лица, а также окклюзии; 4) Классификация признаков лица, в ходе которой происходит сравнение полученных признаков с базой данных из которой выбирается наиболее вероятное совпадение лиц; 5) Выделение целевого диктора и верификация по голосовой парольной фразе; 6) Распознавание голосовой команды.

В ходе работы над приложением для мобильного устройства была реализована система распознавания лица на базе представления признаков лица с использованием обработки фильтром Габора. Изображение лица представляется в виде вейвлета Габора со свёрткой фильтра по ядрам, как описано в работе [12]. Обычный дескриптор лица после обработки фильтром Габора описывается 40 различными признаками, включая 5 различных масштабов изображения и 8 ориентаций. В последующем вейвлет Габора разделяется на несколько непересекающихся фрагментов, которые обрабатываются по методу главных компонент и дискриминантного анализа для уменьшения размерности признаков и формирования конечного дескриптора признаков лица. По опубликованным в статьях результатам использования фильтра Габора совместно с распознаванием по методу главных компонент позволяет достичь точности распознавания 93,8% на наборе данных FERET [13]. Методы распознавания речевых команд и голоса описаны в предыдущих статьях авторов [19] и [18] соответственно.

Портирование описанной алгоритмики по распознаванию лиц с персонального компьютера на экспериментальный стенд и её запуск без использования графического процессора дал результат в 8,5 секунды на поиск и распознавание лица на изображении. На Рис. 6 показано распределение времени выполнения приложения по составляющим алгоритмам. В данной реализации приложения системная функция Android facedetector используется для поиска на изображении областей с лицами, затем AdaBoost-оптимизированная функция по локализации глаз находит опознавательные точки лица. Найденная область лица изменяется в размере до 64x84 пикселя, после чего

Рис. 4. Интерфейс тестового приложения

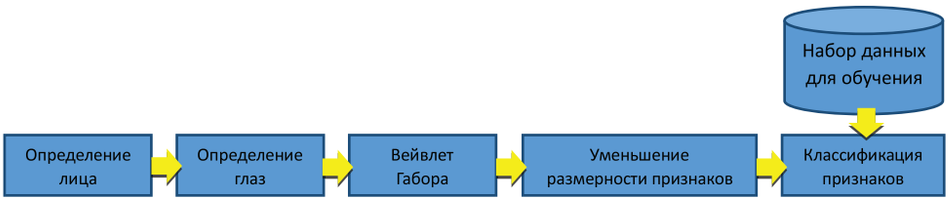

Рис. 5. Схема работы алгоритма распознавания лица

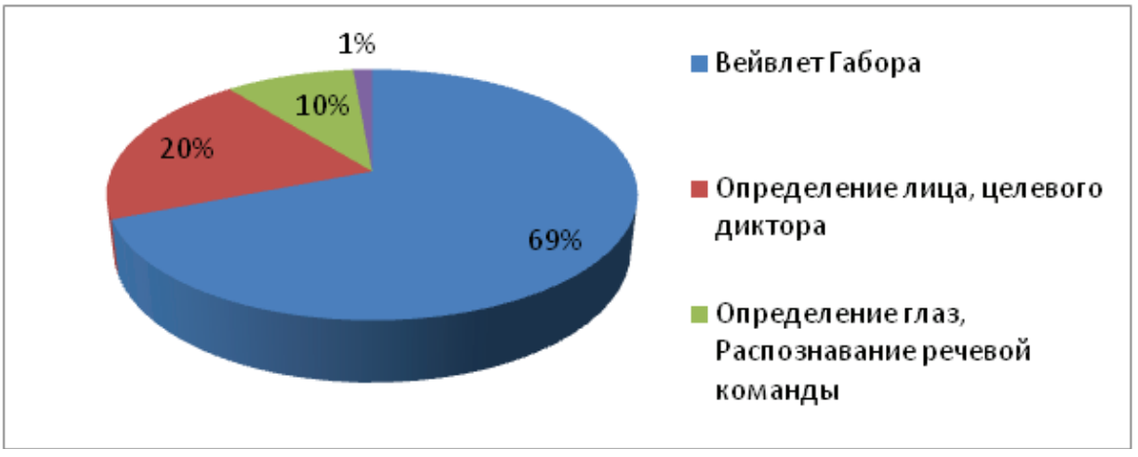

Рис. 6. Распределение процессорного времени на этапы работы алгоритмов распознавания речи, голоса и лица

их неё извлекаются признаки лица. Классификация признаков лица выполняется при помощи метода k ближайших соседей. Длительное время исполнения алгоритмики даёт понять, что задача по распознаванию лица слишком ресурсоёмка для мобильного устройства без использования графического процессора. По полученным данным времени выполнения приложения можно сделать вывод, что для комфортной работы пользователей требуется увеличение скорости исполнения алгоритмики в 5 раз – в особенности это касается вычисления обработки фильтром Габора.

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКОГО ПРОЦЕССОРА ДЛЯ РАСПОЗНАВАНИЯ РЕЧИ, ГОЛОСА И ЛИЦА ЧЕЛОВЕКА НА МОБИЛЬНОМ УСТРОЙСТВА

Вейвлет Габора на графическом процессоре может быть реализован либо свёрткой, либо БПФ. Метод свёртки применим лишь для ядер маленькой размерности из-за ограничения по объёму кэш-памяти графического процессора. Но в любом случае ядра маленькой размерности не подходят для решения задач распознавания образов. Таким образом реализация вейвлета Габора и выделение признаков лица на графическом процессоре сделана на базе метода БПФ, который сначала трансформирует изображение лица и фильтр Габора в фурье-пространство, перемножает их друг на друга, и затем совершает их обратное преобразование в пространственную область.

Экспериментальные результаты сравнения двух реализация одной и той же алгоритмики на центральном и графическом процессорах дана в Таблице 2. Использование графического процессора для вычисления вейвлета Габора позволило уменьшить время вычисления до 1,2 секунды, что означает увеличение быстродействия в 4,25 раз по сравнению с реализацией только на центральном процессоре. При помощи графического процессора, снявшего вычислительную нагрузку с центрального процессора, общее вычислительное время для распознавания 1 человека уменьшилось в 8,5 секунды до 4,6 секунды. Что касается энергопотребления, то связка центрального и графического процессоров на мобильном устройстве потребляет 16,3 Дж, по сравнению

Таблица 2.

Процессорное время и энергопотребление алгоритмов с использованием графического процессора и без него

Функция	Выделение признаков (лицо + речь)		Распознавание лица + речевой команды	
Метод	Центральный процессор	Графический и центральный процессор	Центральный процессор	Графический и центральный процессор
Время (сек)	5,1	1,2	8,5	4,6
Энергия (Дж)	18,7	4,9	29,8	16,3

с 29,8 Дж которые потребляет центральный процессор при вычислении в одиночку. После оптимизации вычисления вейвлета Габора при помощи графического процессора, определение лица в кадре и выделение признаков лица стали наиболее критичными по времени частями алгоритмики. В последующем возможна оптимизация при помощи графического процессора и озвученных вычислительных ограничений.

ЗАКЛЮЧЕНИЕ

В данной статье описано исследование по вычислительной мощности и энергопотреблению центрального и графического процессоров на мобильном устройстве для решения задачи распознавания речи, лица и голоса человека. В описанном в статье примере было реализовано вычисление вейвлета Габора на графическом процессоре мобильного устройства, что позволило поднять производительность в 4,25 раза. Данное экспериментальное исследование подтвердило, что графические процессоры мобильных устройств позволяют значительно увеличить производительность для решения задач распознавания образов, несмотря на то, что изначально данные процессоры разработаны для экономии энергопотребления, а не для высокой производительности. В дополнении к полученному выводу о повышении производительности, можно отметить и совокупное понижение энергопотребления мобильного устройства при использовании графического процессора. Полученные в ходе эксперимента результаты демонстрируют снижение энергопотребления мобильного устройства в 3,98 раза при использовании графического процессора, при незначительном повышении потребляемой мощности.

Благодаря использованию архитектуры с поддержкой сверхмалого энергопотребления и сокращению вычислительных ядер, графические процессоры мобильных устройств потребляют на 2 порядка меньше мощности, чем графические процессоры персональных компьютеров (4 Вт у Qualcomm Snapdragon Adreno против 483 Вт у Nvidia GeForce 8800). Таким образом, графические процессоры мобильных устройств всё равно могут достигать лучшей энергоэффективности при решении задач, несмотря на то, что они намного слабее графических процессоров, используемых в персональных и серверных компьютерах.

Литература

1. Texas Instruments Inc. OMAP3 family of multimedia application processors. URL: http://focus.ti.com

2. Qualcomm Inc.URL: http://www.qualcomm.com/snapdragon

3. Nvidia Corporation, “Bring High-End Graphics to Handheld Devices,” Nvidia whitepaper, 2011.

4. *J. Leskela, J. Nikula, and M. Salmela*, “OpenCL Embedded Profile Prototype in Mobile Device,” IEEE Workshop on Signal Processing Systems, 2009. PP. 279-284.

5. *N. Singhal, I. K. Park, and S. Cho*, “Implementation and Optimization of Image Processing Algorithms on Handheld GPU,” IEEE International Conference on Image Processing, 2010. PP. 4481-4484.

6. *T. Akenine-Möller and J. Ström*, “Graphics Processing Units for Handhelds,” Proceedings of the IEEE, vol. 96, Issue 5, 2008. PP.779-789.

7. *J. Ström and T. Akenine-Möller*, “iPACKMAN: High-Quality, Low-Complexity Texture Compression for Mobile Phones,” in Proc. Graph. Hardware, 2005. PP. 63-70.

8. Khronos Group, OpenGL ES 2.0 Specification. URL: http://www.khronos.org/opengles.
9. Khronos Group, Open Computing Language (OpenCL) Specification. URL: http://www.khronos.org/opencl.
10. Nvidia Corporation, "Nvidia Compute Unified Device Architecture (CUDA) Programming Guide," version 2.0, 2008.
11. *A. Munshi, D. Ginsburg, and D. Shreiner*, "OpenGL ES 2.0 Programming Guide," Addison-Wesley, USA, 2008.
12. *Y. Su, S. Shan, X. Chen, and W. Gao*, "Hierarchical Ensemble of Global and Local Classifiers for Face Recognition," IEEE Transactions on Image Processing, 2009. PP.1885-1896.
13. *P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss*, "The FERET Evaluation Methodology for Race-Recognition Algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000. PP.1090-1104.
14. Open Handset Alliance, Android Software Develop Kit 2.2. URL: http://developer.android.com/intex.html.
15. *T. Sumanaweera and D. Liu*, "Medical image reconstruction with the FFT," In GPU Gems 2, Addison-Wesley, 2005. PP.765-784.
16. *D. Ren and R. Suda*, "Power Efficient Large Matrices Multiplication by Load Scheduling on Multi-core and GPU Platform with CUDA," International Conference on Computational Science and Engineering, Vancouver, August 2009.
17. *L.D. Brandon, C. Boyd, and N. Govindaraju*, "Fast computation of general Fourier Transforms on GPUs," IEEE International Conference on Multimedia and Expo, Hannover, April 2008.
18. *Габдуллин В.В., Капустин А.И., Королев А.И.* «Применение технологии CUDA для задач голосовой биометрии на примере построения универсальной фоновой модели диктора», труды конференции ПаВТ, 2010.
19. *Фирун К.Б., Лапшин П.А., Фролов В.Е.* «Интернет-платформа по работе с аудио-видео информацией с использованием передовых технологий автоматического распознавания речи», труды VII-ой Конференции молодых учёных НИУ ИТМО, 2011.