Литература:

1. Берсини, У. Двойная пластичность иммунной сети как источник инженерных решений // Искусственный иммунные системы. Под ред. Д. Дасгупты. Пер. с англ. Под ред. А.А. Романюхи – М.: ФИЗМАТЛИТ. – 2006. – С. 42–65.
2. Литвиненко, В.И. Применение алгоритма клонального отбора для решения систем алгебраических уравнений / Математичні машини і системи. – № 3. – 2006. – С. 117–126.
3. Литвиненко, В.И. Компьютерная система для решения задач классификации на основе модифицированных иммунных алгоритмов / Дидык А.А., Захарченко Ю.А. // ААЭКС. – №2(22). – 2008.
4. Хант, Д. Jisys: разработка искусственной иммунной системы для практических приложений / Тиммис Д., Кук Д., Нил М., Кинг К. // Искусственный иммунные системы. Под ред. Д. Дасгупты. Пер. с англ. Под ред. А.А. Романюхи – М.: ФИЗМАТЛИТ. – 2006. – С. 188–214.
5. Forrest, S. Self-nonself discrimination in a computer / Perelson A.S., Allen L., Cherikuri R. // In: Proc. Of IEEE symposium on research in security and privacy. – Oakland. – 1994. – P. 202–212.
6. Helman, P. An efficient algorithm for generating random antibody strings / Forrest S. // Technical report №CS94–7. – Department of computer science. – University of New Mexico. –1994.
7. Jerne, N.K. The immune system // Sci. Am. – №1. – 1973. – P. 52–60.

УДК 681.3

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА ВЫЧИТАНИЯ СОСЕДНИХ КАДРОВ НА ОСНОВЕ ТЕХНОЛОГИИ CUDA
PARALLEL IMPLEMENTATION OF ALGORITHM OF FRAMES SUBTRACTION BASED ON TECHNOLOGY CUDA

А.Г. Яшина
A.G. Yashina
ГОУ ВПО «Вятский государственный гуманитарный университет»
Vyatka State University of Humanities

This article presents results of using CUDA technology for parallel computing on GPU for implementation algorithm of subtraction neighboring frames.

Введение

Цифровое видео достаточно давно распространено в современном мире.

Интеллектуальный анализ видеоинформации позволяет извлекать полезные данные для решения задач информационного поиска, автоматизации рабочего процесса, слежения за объектами, распознавания и реагирования на события. Видео является универсальным типом информации, который включает кроме последовательности кадров, также аудио и текстовую информацию (комментарии, заголовки и возможно тэги, описывающие видеофайл). В данной работе видео представляется в виде последовательности кадров, то есть рассматривается только графическая составляющая видеофайла.

Система, выполняющая интеллектуальный анализ, имеет сложную структуру, которая включает множество механизмов и подзадач на разных уровнях. Один из подготовительных этапов для последующего распознавания представлен попиксельной или поблочной обработкой кадров, которая направлена на выделение динамических объектов и фона. Возможным вариантом данной предобработки является алгоритм вычитания соседних кадров [3, 4].

Алгоритм вычитания соседних кадров

Кадр представляется в виде двумерного массива пикселей, являющимися трёхмерными цветовыми векторами, координаты которых соответствуют R, G, B каналам цвета. В результате вычитания соседних кадров будет получен массив векторов, значения которых соответствуют степеням различия соответствующих пикселей. Если данный массив выводить как кадр видео, то динамические объекты будут представляться в виде радужного контура, а статичные – в виде черного фона.

Данный подход неустойчив к шуму, который возникает из-за качества камеры, освещения и условий среды. Одним из способов снижения шума является введение порога для значений результирующих векторов. То есть решение об отношении результирующего вектора к фону принимается в том случае, если значение вектора меньше определённого порога, в противоположном случае считается, что он принадлежит к движущемуся объекту (движущейся области кадра).

Использование «жесткого» порога не учитывает цветовое содержание кадров, что может привести к пропускам некоторых динамических областей, например, когда цвет движущегося объекта близок к цвету фона. Решение данной проблемы возможно путём введения самонастраиваемого порога посредством слежения за разницей между динамичными и статичными объектами и вычислением преобладающих цветов видеопоследовательности.

Технология CUDA

Алгоритм вычитания соседних кадров доступен для распределенной реализации, что должно снизить время его выполнения. В данной работе проводится исследование возможности распараллеливания данного алгоритма на основе технологии CUDA, то есть вычислении разности соседних кадров на видеокарте (GPU), что дополнительно снизит загрузку центрального процессора (CPU).

Видеокарта изначально предназначена для графических вычислений, то есть выполнение однотипных инструкций на большом массиве данных. Технология CUDA [1, 2, 5] от компании NVIDIA представляет аппаратно-программный комплекс для неграфических распределённых вычислений на видеокарте и поддержи-

вается видеокартами от компании NVIDIA, которые являются SIMD-устройствами (Single Instruction, Multiple Data). Вычисления производятся в блоке обработки текстур (Texture processing cluster), который в свою очередь, содержит потоковые мультипроцессоры (Streaming multiprocessor), включающие потоковые процессоры (Streaming processor, на которых в основном происходят CUDA-вычисления), модули для сложных вычислений (Special function unit) и общую память (Shared memory). Количество процессоров видеокарты зависит от её серии, например, GPU GeForce 8500GT и GeForce 8600GT имеют 16 CUDA-процессоров, а GPU GeForce 8800GT – 112.

Технология CUDA хорошо подходит для организации параллельного вычисления разности кадров, так как данные независимы между собой, легко разделимы на блоки для вычислений в отдельной нити и не требуют сложной обработки.

Реализация приложения

Приложение, реализующее алгоритм вычитания соседних кадров, написано на платформе .Net, на языке C# и построено на основе объектно-ориентированного подхода. Для CUDA вычислений используется библиотека CUDA.NET, предоставляющая средства подключения CUDA-модуля. Для захвата кадров видео файла или веб-камеры используется библиотека EmguCV, представляющая .Net обёртку OpenCV.

Кадр представляется в виде экземпляра класса Bitmap, который затем преобразовывается в массив byte посредством стандартного метода System.Runtime.InteropServices.Marshal.Copy(). Работа приложения по обработке видеопоследовательности происходит в режиме реального времени.

Результаты эксперимента

Эксперимент проводился на видео разрешения 640x480, то есть длина массива данных составляла 640*480*4=1228800 элементов (R, G, B каналы цвета и альфа канал). Данные делятся на блоки по 524288 элементов, и каждый блок выполняется на 512 нитях.

В ходе эксперимента показатели вычислений на видеокарте (NVIDIA 8600GT) сравнивались с показателями вычислений организованных в виде цикла вычитания двух массивов на процессоре AMD Athlon X2 Dual 2.11 ГГц. Замеры времени выполнения осуществлялись посредством класса Stopwatch пространства имён System.Diagnostics.

Показатели эксперимента приведены в Таблице 1. Соотношение среднего времени выполнения показаны на Рис. 1.

Данные эксперимента показывают улучшение времени выполнения алгоритма вычитания соседних кадров при его распараллеливании на основе технологии CUDA.

Используя средние значения времени выполнения алгоритма и формулу

$$k = \frac{T_{CPU}}{T_{GPU}},$$

где T_{CPU} – время выполнения алгоритма на CPU и T_{GPU} – время выпол-

№ кадра	GPU мс.	CPU мс	№ кадра	GPU мс	CPU мс	№ кадра	GPU мс	CPU мс	№ кадра	GPU мс	CPU мс
1	19	25	13	17	30	71	15	32	83	15	29
2	17	27	14	16	26	72	16	28	84	17	29
3	16	34	15	16	27	73	15	28	85	16	32
4	17	28	16	16	25	74	15	29	86	16	27
5	16	32	17	17	28	75	17	29	87	17	26
6	16	31	18	15	27	76	15	29	88	19	27
7	28	29	19	18	26	77	17	28	89	16	29
8	17	28	20	16	27	78	15	27	90	16	27
9	16	31	21	15	27	79	15	30	91	17	29
10	15	30	22	17	30	80	15	29	92	17	27
11	14	31	23	16	31	81	16	29	93	16	66
12	16	30	24	15	30	82	20	28	94	15	28

Таблица 1. Время выполнения алгоритма

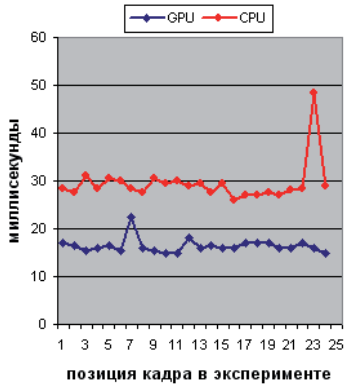

Рис. 1 Среднее время выполнения

нения алгоритма на GPU, получаем коэффициент отношения времени выполнения алгоритма на CPU и GPU

$$k = \frac{29,4167}{16,4375} = 1.7896 \cdot$$

Заключение

В современных приложениях обработки видео широко распространяется применение технологий распределенных вычислений. Создаются различные параллельные алгоритмы сжатия, определения движения, распознавания образов и другие вспомогательные алгоритмы для обработки видео.

Таким образом, результаты эксперимента показали, что использование технологии CUDA является эффективным при обработке кадров видеопоследовательности и перспективным в данной области вычислений.

Литература:

1. CUDA.NET для .NET-разработчика [Электронный ресурс] – http://www.thevista.ru/page.php?id=13082

2. NVIDIA CUDA Programming Guide [Электронный ресурс] – http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf

3. Гаганов В. Сегментация движущихся объектов в видеопотоке [Электронный ресурс] / Гаганов В., Конушин А. // Компьютерная графика и мультимедиа (Сетевой журнал), 2004. – http://cgm.computergraphics.ru/content/view/67

4. Форсайт Д., Компьютерное зрение [Текст] / Понс Ж. // – Москва Санкт-Петербург Киев, 2004.

5. Фролов В., Введение в технологию CUDA [Электронный ресурс] // – 2008. – http://cgm.computergraphics.ru/issues/issue16/cuda