13. Arshinsky V.L. Metodicheskiy podkhod k sobytiynomu modelirovaniyu v issledovaniyakh energeticheskoy bezopasnosti [Methodical approach to event simulation in energy security research]. *Informatsionnye i matematicheskie tekhnologii v nauke i upravlenii. Trudy XV Baykalskoy Vserossiyskoy konferentsii* [Proc. XV Baikal Russian Conference. Mathematical and Informational Technologies in Science and Management]. Irkutsk, ESI SB RAS, 2010. P. III, pp. 120–129.
14. Axelrod R. *Structure of decision*. Princeton, New Jersey, Princeton University Press, 1976. 404 p.
15. Novik K.V. *Set avtomatov dlya modelirovaniya asinkhronnogo vzaimodeystviya protsessov. Avtoreferat dis. kand. nauk* [Network automats for modelling asynchronous communication processes. Abstract. Cand diss.]. Moscow, 2006. 22 p.
16. Stolyarov L.N., Novik K.V. Joiner-set dlya modelirovaniya vzaimodeystvuyushchikh parallelnykh protsessov [Joiner-interacting network for modelling concurrent processes]. *Informatsionnye i matematicheskie tekhnologii. Sbornik nauchnykh trudov* [Information and mathematical techniques. Scientific papers]. Moscow, Moscow Institute of Physics and Technology Press, 2004. pp. 81–97.
17. Stolyarov L.N., Novik K.V. Realizatsiya parallelnykh protsessov s pomoshchyu setey Joiner-net [Implementation of parallel processes with a help of Joiner-net]. *Informatsionnye i matematicheskie tekhnologii Sbornik nauchnykh trudov* [Informational and Mathematical technologies. Scientific papers]. Irkutsk, ESI SB RAS, 2004. pp. 11–14.
18. Anisimov M.M. Upravlenie sobytiynymi setyami [Event-management networks]. *Informatsionnye i matematicheskie tekhnologii v nauke i upravlenii. Trudy XIV Baykalskoy Vserossiyskoy konferentsii* [Proc. XIV Baikal All Russia Conference. Information technology and mathematical science and management]. Irkutsk, ESI SB RAS, 2009. P. 3, pp. 238–240.

УДК 004.9

НЕЙРОСЕТЕВОЙ ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ СЛЕЖЕНИЯ ЗА ОБЪЕКТОМ В РЕАЛЬНОМ ВРЕМЕНИ

Тарков Михаил Сергеевич,
канд. техн. наук, ст. науч. сотр. лаборатории физических основ интегральной микроэлектроники Института физики полупроводников им. А.В. Ржанова СО РАН, Россия, 630090, Новосибирск, пр. Ак. Лаврентьева, 13. E-mail: tarkov@isp.nsc.ru

Дубынин Сергей Владимирович,
магистрант Новосибирского государственного университета, Россия, 630090, г. Новосибирск, ул. Пирогова, 2. E-mail: dubyninsergey@gmail.com

Актуальность исследования обусловлена необходимостью разработки программных средств слежения за объектами в реальном масштабе времени.

Цель работы: *Создание алгоритма слежения за объектом в кадре в реальном масштабе времени.*

Методы исследования: *Параллельная реализация сигмоидальной нейронной сети на графическом процессоре, замеры временных характеристик параллельного алгоритма и его оптимизация.*

Результаты: *Предложена реализация на графическом процессоре (GPU) нейросетевого алгоритма слежения за объектом, спецификой которого является использование при обучении нейронной сети задачника, устанавливающего однозначное соответствие обрабатываемого кадра в видеопотоке координатам центра объекта в кадре. Благодаря использованию GPU удается решить задачу слежения в реальном масштабе времени (25 кадров в секунду) при размерах обрабатываемого кадра до 1280×960. Алгоритм основан на использовании многослойного персептрона и имеет ряд параметров, которые определены экспериментально. Одним из таких параметров является число нейронов скрытого слоя. В связи с реализацией алгоритма на GPU рассмотрены числа нейронов, кратные 16. В экспериментах установлено, что 16 и 32 нейрона не могут обеспечить даже малой степени запоминания образов, 48 нейронов справлялись с обучением только на малых обучающих выборках, 64 нейрона обеспечили хорошую степень запоминания образов и скорость работы. Дальнейшее увеличение числа нейронов приводит только к уменьшению скорости работы нейронной сети и ее обучения.*

Также заслуживает внимания частота, с которой нужно брать кадры из видеозаписи, чтобы эффективно обучить нейронную сеть. Экспериментально установлено, что на частоте выборки одного кадра из десяти сумма максимальных отклонений по обеим координатам равна 50 при размерах объекта 300×300; дальнейшее увеличение частоты кадров лишь замедляет процесс обучения, не давая существенного выигрыша в качестве.

Получены ускорения процесса слежения в 10 раз по сравнению с центральным процессором персонального компьютера. Процесс обучения нейронной сети ускорился в среднем только в 2 раза. Это обусловлено необходимостью транспонирования матриц весов при реализации обучения нейронной сети на GPU.

Для реализации параллельного алгоритма использована программно-аппаратная архитектура CUDA, позволяющая производить вычисления с использованием графических процессоров NVIDIA, поддерживающих технологию GPGPU (произвольных вычислений на видеокартах). Для предварительной обработки изображений и вывода информации использовалась библиотека компьютерного зрения OpenCV.

Ключевые слова:
Слежение за объектом, нейронная сеть, параллельные вычисления, графический процессор, CUDA.

Введение

В настоящее время разработано большое количество приложений для видеообработки с использованием графических процессоров [1–9], которые обеспечивают массовый параллелизм обработки данных в реальном масштабе времени. Одной из самых распространенных задач в этой сфере является слежение за объектами [3–10]. Использование алгоритмов слежения за объектами используется для различных целей: выявление определенных движущихся целей и слежение за ними в военной технике; фиксация номерных знаков автомобилей, превышающих скорость; наложение различных визуальных эффектов на видеозапись и прочее.

Для реализации слежения за объектами разработано множество методов и алгоритмов, но зачастую они являются узкоспециализированными и устойчивы лишь на определенном типе видеозаписей. В хороших условиях (при четких изображениях, при низкой скорости перемещения объекта и прочее) эти алгоритмы работают достаточно хорошо, но при возникновении помех, при увеличении скорости объекта и уменьшении его размеров, алгоритмы дают сбои. Ко всему прочему, алгоритмы слежения за объектами являются достаточно трудоемкими, что вынуждает сжимать обрабатываемый кадр или как-то иначе упрощать обрабатываемую информацию. В связи с этим и возникает проблема разработки эффективных робастных алгоритмов слежения за объектами.

В решении задачи слежения на графических процессорах активно используются нейронные сети [6–8] и другие математические модели [4, 5, 9]. В данной работе предложена реализация на графическом процессоре нейросетевого алгоритма слежения за объектом [10], спецификой которого является использование при обучении нейронной сети задачника, задающего однозначное соответствие обрабатываемого кадра в видеопотоке координатам центра объекта в кадре. Благодаря использованию графического ускорителя удается решить задачу слежения в реальном масштабе времени без уменьшения размеров обрабатываемого кадра.

Постановка задачи

Существует множество различных систем слежения за объектами. Эти системы используют разные алгоритмы и работают на различных входных данных. Наиболее эффективные реализации используют сложное дорогостоящее оборудование: несколько видеокамер, запись цветного видео или видео в инфракрасном спектре. С одной стороны, цветное изображение дает возможность использовать больше различных алгоритмов, но, с другой стороны, эти алгоритмы достаточно трудоемки и не всегда могут корректно работать (например, при слабом освещении). Алгоритмы для монохромных изображений могут использовать более доступную технику, но являются менее эффективными и зачастую используют небольшие разрешения кадра.

Алгоритм [10] работает с монохромными изображениями малого разрешения (320×240), которое во время предобработки данных снижается до 80×60. Однако можно создать быстрый алгоритм, работающий с большими разрешениями кадра в реальном масштабе времени на достаточно дешевом оборудовании. Ключ к такому решению лежит в использовании графических карт как устройств, позволяющих выполнять массивно-параллельные вычисления [11, 12].

Общая задача заключается в слежении за объектом на изображениях и видео, то есть в определении координат центра объекта на основе информации, получаемой из изображения. Так как одной из целей работы является слежение за объектом на видео в реальном масштабе времени, то на алгоритм определения координат центра объекта накладывается ограничение по быстродействию: на обработку одного кадра должно уходить не более 1/25 секунды. При этом требуется обрабатывать кадр целиком без потерь информации. Отметим, что в [10] для повышения скорости работы алгоритма использовался лишь каждый четвертый пиксель изображения.

Нейронная сеть и её обучение

Для обработки изображений использовалась сигмоидальная сеть прямого распространения с одним скрытым слоем [13–18]. Искусственная нейронная сеть представляет собой систему взаимосвязанных простых процессоров – нейронов. Каждый нейрон получает входные сигналы w_i, $i=0,1,\ldots,N$ и порождает выходной сигнал $y=f(u)$, где $f(u)$ – нелинейная функция активации; $u=\sum_{i=0}^{N} w_i x_i$ – активация нейрона, w_i, $i=0,1,\ldots,N$ – весовые коэффициенты нейрона, w_0 – величина порога, $x_0 \equiv 1$.

Обучение нейрона заключается в выборе весовых коэффициентов w_i таким образом, что выходной сигнал y совпадает с требуемой величиной d. Обучение с учителем использует набор обучающих примеров, то есть множество пар вида (x,d), где x – вектор входных сигналов.

Для униполярного сигмоидального нейрона функция активации задана выражением

$$f(u)=\frac{1}{1+\exp(-\beta u)},$$

где β – параметр функции активации. В трехслойной сигмоидальной сети, первый слой нейронов содержит входные сигналы, скрытый слой нейронов получает входные сигналы и преобразует их, чтобы передать выходному слою нейронов. Выходной слой активизируется и порождает выходные сигналы нейронной сети.

Благодаря дифференцируемости функции активации для обучения нейронной сети можно использовать градиентные методы оптимизации. В частности используется метод наискорейшего

спуска (метод обратного распространения ошибки), согласно которому модификация вектора весов производится в направлении отрицательного градиента целевой функции

$$E(w) = \frac{(y-d)^2}{2},$$

то есть

$$w(t+1) = w(t) - \alpha \nabla E(w(t)),$$

где $\alpha \in (0,1]$ – коэффициент (шаг) обучения.

Перед обработкой производилась нормировка яркости монохромных изображений в диапазон [0,1]. В результате исследования поведения сети при разном числе нейронов в скрытом слое принято решение использовать 64 нейрона, что обеспечивает достаточную скорость работы алгоритма и его точность. В выходном слое присутствует всего два нейрона, каждый из которых дает на выходе одну из координат искомого объекта. Для обучения использовался алгоритм обратного распространения ошибки с коррекцией величины шага. В качестве задачника использовался набор изображений с известными координатами центра объекта.

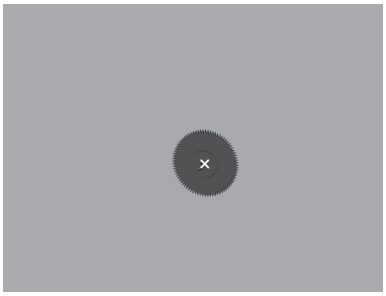


Рис. 1. *Пример обучающего изображения*

Для создания задачника использовалась программа Autodesk Maya 2011 [19], в которой создана трехмерная модель шестеренки (рис. 1).

Для этой модели с помощью формул, зависящих от номера кадра, заданы следующие параметры: координаты объекта в кадре, углы поворота объекта по трем осям и размер объекта. Затем с помощью программы получены изображения с объектом и видеозапись для тестирования процесса слежения за объектом. Для создания файла с координатами центра объекта на разных изображениях создана программа, вычисляющая координаты центра по формулам, описывающим его движение, и записывающая эти координаты в файл.

Для реализации параллельного алгоритма использована программно-аппаратная архитектура CUDA [11, 12], позволяющая производить вычисления с использованием графических процессоров NVIDIA, поддерживающих технологию GPGPU (произвольных вычислений на видеокартах). Для предварительной обработки изображений и вывода информации по ходу обучения использовалась библиотека компьютерного зрения OpenCV [20].

Реализация нейронной сети на основе технологии CUDA

CUDA (Compute Unified Device Architecture) – это интегрированная среда, позволяющая разрабатывать программы на языке C/C++, которые запускают параллельное исполнение специальных функций ядра на графической карте, поддерживающей технологию CUDA. Эта графическая карта в данном контексте называется устройством, а компьютер, на котором установлено устройство, называется хостом (host). Функции ядра выполняются параллельно нитями, которые объединяются в блоки одинакового размера. Блоки и нити внутри блоков формируют сетку, которая в ходе выполнения функции отображается на мультипроцессоры GPU (Graphic Processing Unit) и их (скалярные) процессоры, соответственно.

Функции ядра могут использовать различные типы памяти устройства: регистры, разделяемая (shared), текстурная и константная памяти. Текстурная и константная памяти являются малыми по объему, но быстрыми памятями устройства. Локальная и глобальная памяти устройства работают значительно медленнее, но вмещают значительно больше данных (обычно до 2-х Гбайт). Регистры и локальная память доступны только текущей нити, разделенная память доступна каждому блоку, а константная, текстурная и глобальная памяти доступны всем нитям. Передача данных между блоками возможна только через глобальную память.

Поскольку нейроны одного слоя могут выполнять вычисления независимо друг от друга, принято решение реализовать параллельные версии основных функций нейронной сети и подстройки ее весов. Созданы дополнительные функции, обеспечивающие взаимосвязь параллельного алгоритма с интерфейсом и последовательной версией. Например, программа позволяет обучить нейронную сеть с использованием CUDA, а обрабатывать выбранные пользователем изображения с помощью последовательной реализации алгоритма и наоборот.

Функцию, выполняемую нейронной сетью, можно разбить на три части:

1) Входной вектор x умножить на матрицу весов скрытого слоя W_1, после чего к результату прибавить вектор смещений скрытого слоя b_1:
$$a_1 = W_1 \cdot x + b_1. \quad (1)$$
2) К полученному вектору a_1 применить функцию активации f:
$$u = f(a_1). \quad (2)$$
3) Полученный вектор u умножить на матрицу весов выходного слоя W_2 и прибавить вектор смещений b_2 нейронов выходного слоя:
$$y = W_2 \cdot u + b_2. \quad (3)$$

В результате получаем три процедуры, содержащие большое число операций, которые можно выполнить параллельно. Для этих процедур написаны функции ядра GPU, которые запускаются параллельно на множестве нитей.

В основе функций (1) и (3) лежит операция произведения матрицы на вектор. Функция (2) проста и не требует оптимизации. В изначальной версии реализации функции (1) каждая нить производит перемножение одной строки матрицы W_1 на вектор x. Число нитей фиксировано и соответствует числу нейронов скрытого слоя. Каждая нить выполняет операции умножения и сложения. Обращения нитей к глобальной памяти GPU значительно замедляют их работу. К тому же профилировщик Compute Visual Profiler показал, что при этом не используется свойство GPU, позволяющее объединять запросы к памяти: когда следующие друг за другом нити обращаются в следующие друг за другом ячейки памяти, эти обращения могут быть объединены в одно (warp) и вместо группы обращений происходит по сути одно. Максимальное число нитей, входящих в warp, равно 16 (32 для более новых моделей GPU).

Чтобы устранить указанные недостатки, решено:

1) Увеличить число нитей, чтобы каждая нить перемножала только векторы из 16 чисел. В результате увеличивается полезная нагрузка на GPU, больше нитей выполняется параллельно, появляется необходимость новой функции ядра, суммирующей результаты работы нитей.
2) Использовать вместо глобальной памяти разделяемую память (shared memory). Эта память выделяется каждому блоку нитей и может использоваться всеми нитями блока. Для этого надо загрузить фрагмент глобальной памяти, используемый всеми нитями блока, в разделяемую память. Каждая нить делает лишь одно обращение к соответствующей ячейке глобальной памяти, скопировав значение в разделяемую память, а остальные данные нить сможет получить из разделяемой памяти.
3) Транспонировать матрицу W_1, чтобы обращения к элементам матрицы, находящимся в глобальной памяти, объединялись в warp. Изначально одна нить работала с вектором-строкой, элементы которой располагаются в разных сегментах памяти. Транспонирование матрицы позволяет нити работать с вектором-столбцом, что объединяет подряд идущие нити в warp.

В результате этих модификаций суммарное время выполнения функции ядра для первого блока по всем запускам в течение работы программы уменьшилось с 30 % времени работы GPU до 1 % плюс 3 % на суммирующую функцию ядра, появившуюся в ходе изменений пункта 1. Все загрузки из глобальной памяти и выгрузки в глобальную память GPU являются объединенными (coalesced). Ветвления, вызванные циклом функции, и запуск блоков нитей (warps) занимают крайне мало времени.

В функции подстройки весов определенные блоки операций организованы в функции ядра, выполняющиеся параллельно на множестве нитей. Однако специфика этих блоков не позволяет сильно ускорить их и вынуждает использовать транспонирование матриц перед вызовом функции подстройки весов и после нее, чтобы на следующей итерации функция работы нейронной сети получила транспонированные матрицы весов. Без транспонирования наблюдается замедление работы функции подстройки весов.

Алгоритм имеет ряд параметров, которые определены экспериментально. Одним из таких параметров является число нейронов скрытого слоя. В связи со спецификой параллельной реализации алгоритма рассмотрены числа нейронов, кратные 16 (размер warp равен 16). В экспериментах установлено, что 16 и 32 нейрона не могут обеспечить даже малой степени запоминания образов, 48 нейронов справлялись с обучением только на малых обучающих выборках, 64 нейрона обеспечили хорошую степень запоминания образов и скорость работы. Дальнейшее увеличение числа нейронов приводит только к уменьшению скорости работы нейронной сети и ее обучения.

Кроме вышеуказанных параметров, внимания также заслуживает частота, с которой нужно брать кадры из видеозаписи, чтобы эффективно обучить нейронную сеть. Экспериментально установлено (рис. 2), что:

1) на частоте выборки одного кадра из десяти сумма максимальных отклонений по обеим координатам равна 50 при размерах объекта 300×300;
2) дальнейшее увеличение частоты кадров лишь замедляет процесс обучения, не давая существенного выигрыша в качестве.

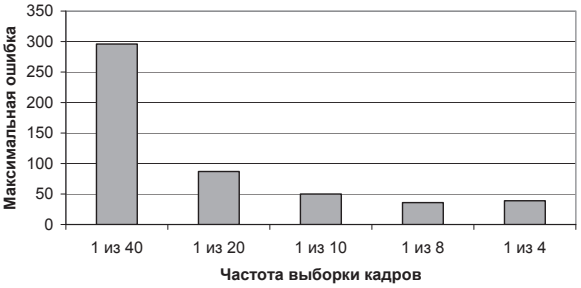
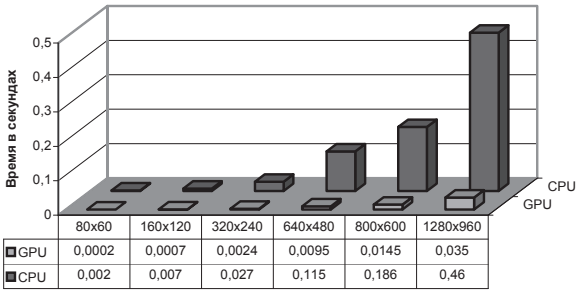


Рис. 2. *Зависимость ошибки определения координат объекта от частоты выборки кадров*

Тестирование параллельной и последовательной реализаций проводилось на компьютере со следующими характеристиками: CPU – AMD Athlon 7750, 2 ядра по 2,7 ГГц, GPU – NVIDIA GeForce 9800 GT. 512MB 256 bit, количество потоковых процессоров – 112. Разработка параллельной версии велась с использованием Cuda Toolkit 4.1. и Cuda Toolkit 4.0 (использован профилировщик этой версии). Главным параметром, по которому проводилось сравнение, является скорость работы нейронной сети, то есть быстродействие основной функции слежения за объектом.

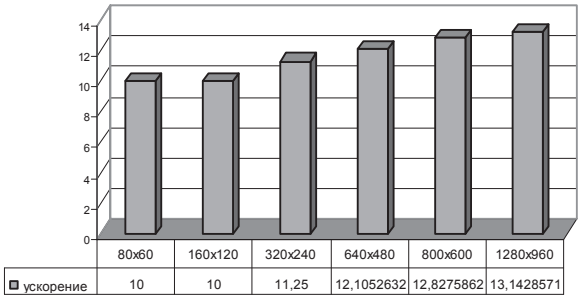
Из рис. 3 следует, что параллельная реализация нейронной сети на GPU позволяет увеличить линейные размеры обрабатываемых изображений

в 4 раза (с 320×240 до 1280×960). Из рис. 3, 4 следует, что процесс обработки нейронной сетью последовательности кадров ускорился в среднем в 10 раз. Процесс обучения ускорился в среднем только в 2 раза (рис. 5). Это обусловлено необходимостью транспонирования матриц весов при реализации обучения нейронной сети на GPU.



	80x60	160x120	320x240	640x480	800x600	1280x960
GPU	0,0002	0,0007	0,0024	0,0095	0,0145	0,035
CPU	0,002	0,007	0,027	0,115	0,186	0,46

Рис. 3. *Времена работы нейронной сети на CPU и с использованием GPU*



	80x60	160x120	320x240	640x480	800x600	1280x960
ускорение	10	10	11,25	12,1052632	12,8275862	13,1428571

Рис. 4. *Ускорение параллельной реализации алгоритма по сравнению с последовательной*

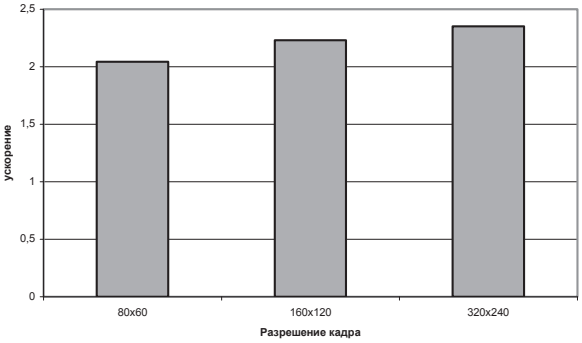


Рис. 5. *Ускорение обучения нейронной сети при различных разрешениях кадра*

Заключение

Реализованы алгоритмы слежения за объектами в реальном масштабе времени, основанные на нейронных сетях с обучением алгоритмом обратного распространения ошибки. Предложен алгоритм слежения за объектами, использующий массивно-параллельные вычисления с использованием графического процессора, проведены исследования и оптимизация параметров алгоритмов. Получены ускорения процесса слежения в 10 раз и процесса обучения в 2 раза. Экспериментально определены максимальные разрешения кадра, пригодные для слежения в реальном времени, и оптимальная частота взятия кадров из видеозаписи в обучающую выборку. Возможно развитие работы в направлении создания алгоритмов, обучающихся в реальном времени, то есть уже во время слежения за объектом.

СПИСОК ЛИТЕРАТУРЫ

1. Real-time image segmentation on a GPU / A. Abramov, T. Kulvicius, F. Worgotter, B. Dellen // Facing the Multicore-Challenge. Lecture Notes in Computer Science. – Berlin: Springer, 2010. – V. 6310. – P. 131–142.
2. NeuFlow: Dataflow Vision Processing System-on-a-Chip / P.-H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, E. Culurcielo // Proc. of 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS). – West Lafayette, IN, USA, 5–8 Aug., 2012. – P. 1044–1047.
3. Di Salvo R., Pino C. Image and Video Processing on CUDA: State of the Art and Future Directions // MACMESE'11: Proc. of the XIII WSEAS International conference on Mathematical and computational methods in science and engineering. – Catania, Italy, 2011. – P. 60–66.
4. Fully automatic extraction of salient objects from videos in near real-time / K. Akamine, K. Fukuchi, A. Kimura, S. Takagi // Computer Vision and Pattern Recognition. – 2010. – C. 1–23. URL: http://arxiv.org/pdf/1008.0502.pdf (дата обращения: 30.03.2014).
5. Real-time GPU color-based segmentation of football players / M.A.M. Laborda, E.F.T. Moreno, J.M. del Rincon, J.E.H. Jaraba // Journal of Real-Time Image Processing. – 2012. – V. 7. – Iss. 4. – P. 267–279.
6. OpenCL Implementation of a Color Based Object Tracking / M. Jocić, D. Obradović, Z. Konjović, D. Tertei // Proc. of III International Conference on Information Society Technology (ICIST). – Novi Sad, Serbia, 2013. – P. 7–11.
7. Ugolotti R., Nashed Y.S.G., Cagnoni S. Real-Time GPU Based Road Sign Detection and Classification // Proc. of PPSN. Lecture Notes in Computer Science. – Berlin: Springer, 2012. – V. 7491. – P. I. – P. 153–162.
8. Pleshkova S. Spiking Neural Networks for Real-Time Infrared Images Processing in Thermo Vision Systems // Recent Researches in Circuits and Systems. – 2012. – P. 183–187. URL: http://hgpu.org (дата обращения: 30.03.2014).
9. Ferreira J.F., Lobo J., Dias J. Bayesian real-time perception algorithms on GPU. Real-time implementation of Bayesian models for multimodal perception using CUDA // Journal of Real-Time Image Processing. – 2011. – V. 6. – Iss. 3. – P. 171–186.
10. Design and Implementation of a Neural Network for Real-Time Object Tracking / J. Ahmed, M.N. Jafri, J. Ahmad, M.I. Khan // World Academy of Science, Engineering and Technology. – 2005. – № 6. – P. 209–212.
11. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.
12. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров. – М.: ДМК Пресс, 2011. – 232 с.
13. Осовский С. Нейронные сети для обработки информации. – М.: Финансы и статистика, 2002. – 344 с.
14. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия – Телеком, 2004. – 452 с.
15. Хайкин С. Нейронные сети: полный курс. – М.: Вильямс, 2006. – 1104 с.

16. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. – Новосибирск: Наука, 1996. – 276 с.
17. Галушкин А.И. Нейронные сети: основы теории. – М.: Горячая линия – Телеком, 2010. – 496 с.
18. Тарков М.С. Нейрокомпьютерные системы. – М.: Интернет-Ун-т Информ. Технологий: Бином. Лаборатория знаний, 2006. – 142 с.
19. Maya – 3D Animation – Autodesk. URL: http://usa.autodesk.com/maya/ (дата обращения: 30.03.2014).
20. Bradski G., Kaehler A. Learning OpenCV. – USA, Sebastopol, CA: O'Reilly Media, Inc., 2008. – 556 p.

Поступила 31.03.2014 г.

UDC 004.9

NEURAL NETWORK PARALLEL ALGORITHM FOR REAL-TIME OBJECT TRACKING

Mikhail S. Tarkov,
Cand. Sc., Laboratory of Physical Foundations of Integrated Microelectronics,
A.V. Rzhanov Institute of Semiconductor Physics SB RAS, 13, Lavrentiev avenue,
Novosibirsk, 630090, Russia. E-mail: tarkov@isp.nsc.ru

Sergey V. Dubynin,
Novosibirsk State University, 2, Pirogov street, Novosibirsk,
630090, Russia. E-mail: dubyninsergey@gmail.com

The urgency of the discussed issue is caused by the need to provide software for tracking objects in real time.
The main aim of the study: *to create an object-tracking algorithm in the frame in real time.*
The methods used in the study: *parallel implementation of the sigmoid neural network on the GPU, measuring the temporal characteristics of the parallel algorithm and its optimization.*
The results: *The authors have proposed implementation of a neural network algorithm on graphic processor (GPU) for tracking an object in a video frame. The specific character of the algorithm is the use of a training set which establish correspondence between the video frame and the object center coordinates in this frame when training a neural network. Owing to GPU application the tracking problem can be solved in real time (25 frames per second) at the processed frame sizes up to 1280×960.*
The algorithm is based on the use of multilayer perceptron and has a number of parameters, which are determined experimentally. One of such parameters is the number of the hidden layer neurons. Due to the algorithm implementation on GPU the authors considered the number of neurons multiple 16. It was determined experimentally that 16 and 32 neurons cannot provide even a small degree of memorizing images, 48 neurons cope with learning only small training samples, and 64 neurons provided a good degree of memorizing images and speed. Further increase in the number of neurons results only in reducing speed of the neural network functioning and its training.
The frequency which is required for taking pictures from a video to train effectively a neural network is worth noticing as well. It is found out experimentally that at a sampling rate of one frame of ten, the sum of the maximum deviations in coordinates is 50, when the object size is 300×300; further increase of the frame rate slows down the process of training without significant gain in quality.
The authors obtained the tracking accelerating by 10 times in comparison with the CPU of a personal computer. The neural network training is accelerated only 2 times on average. This is caused by the need to transpose the weight matrices when implementing the neural network training on the GPU.
To implement the parallel algorithm, the hardware and software architecture CUDA is used. It allows computation on graphics processors NVIDIA, supporting GPGPU technology (general purpose computations on GPU). For preliminary image processing and data output the computer vision library OpenCV is used.

Key words:
Object tracking, neural network, parallel computing, GPU, CUDA.

REFERENCES

1. Abramov A., Kulvicius T., Worgotter F., Dellen B. Real-time image segmentation on a GPU. *Lecture Notes in Computer Science*. Berlin, Springer, 2010, vol. 6310, pp. 131–142.
2. Pham P.-H., Jelaca D., Farabet C., Martini B., LeCun Y., Culurciello E. NeuFlow: Dataflow Vision Processing System-on-a-Chip. *Proc. of 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*. West Lafayette, IN, USA, 2012, 5–8 Aug., 2012, pp. 1044–1047.
3. Di Salvo R., Pino C. Image and Video Processing on CUDA: State of the Art and Future Directions. MACMESE'11: *Proc. of the XII WSEAS International conference on Mathematical and computational methods in science and engineering*. Catania, Italy, 2011, pp. 60–66.
4. Akamine K., Fukuchi K., Kimura A., Takagi S. Fully automatic extraction of salient objects from videos in near real-time. *Computer Vision and Pattern* Recognition, 2010, pp. 1–23. Available at: http://arxiv.org/pdf/1008.0502.pdf (accessed 30 March 2014).
5. Laborda M.A.M., Moreno E.F.T., Del Rincon J.M., Jaraba J.E.H. Real-time GPU color-based segmentation of football players. *Journal of Real-Time Image Processing*, 2012, vol. 7, Iss. 4, pp. 267–279.
6. Jocić M., Obradović D., Konjović Z., Tertei D. OpenCL Implementation of a Color Based Object Tracking. *Proc. of III International Conference on Information Society Technology (ICIST)*. Novi Sad, Serbia, 2013, pp. 7–11.
7. Ugolotti R., Nashed Y.S.G., Cagnoni S. Real-Time GPU Based Road Sign Detection and Classification. *Proc. of PPSN. Lecture*

Notes in Computer Science. Berlin, 2012, vol. 7491, pp. 153–162.
8. Pleshkova S. Spiking Neural Networks for Real-Time Infrared Images Processing in Thermo Vision Systems. *Recent Researches in Circuits and Systems*, 2012, pp. 183–187. Available at: http://hgpu.org (accessed 30 March 2014).
9. Ferreira J.F., Lobo J., Dias J. Bayesian real-time perception algorithms on GPU. Real-time implementation of Bayesian models for multimodal perception using CUDA. *Journal of Real-Time Image* Processing, 2011, vol. 6, Iss. 3, pp. 171–186.
10. Ahmed J., Jafri M.N., Ahmad J., Khan M.I. Design and Implementation of a Neural Network for Real-Time Object Tracking. *World Academy of Science, Engineering and* Technology, 2005, no. 6, pp. 209–212.
11. Boreskov A.V., Kharlamov A.A. *Osnovy raboty s tekhnologiey CUDA* [The basics of working with technology CUDA]. Moscow, DMK Press, 2010. 232 p.
12. Sanders Dzh., Kendrot E. *Tekhnologiya CUDA v primerakh: vvedenie v programmirovanie graficheskikh protsessorov* [CUDA by Example: An introduction to general-purpose GPU programming]. Moscow, DMK Press, 2011. 232 p.
13. Osovskiy S. *Neyronnye seti dlya obrabotki informatsii* [Neural networks for information processing]. Moscow, Finansy i statistika, 2002. 344 p.
14. Rutkovskaya D., Pilinskiy M., Rutkovskiy L. *Neyronnye seti, geneticheskie algoritmy i nechetkie sistemy* [Neural networks, genetic algorithms and fuzzy systems]. Moscow, Goryachaya liniya – Telekom, 2004. 452 p.
15. Khaykin S. *Neyronnye seti: polny kurs* [Neural networks: A comprehensive foundation]. Moscow, Vilyams Publ., 2006. 1104 p.
16. Gorban A.N., Rossiev D.A. *Neironnye seti na personalnom compyutere* [Neural networks on PC]. Novosibirsk, Nauka Publ., 1996. 276 p.
17. Galushkin A.I. *Neyronnye seti: osnovy teorii* [Neural networks: basic theory]. Moscow, Goryachaya liniya – Telekom, 2010. 496 p.
18. Tarkov M.S. *Neyrokompyuternye sistemy* [Neurocomputer systems]. Moscow, Internet Universitet Informatsionnykh tekhnologiy, Binom. Laboratoriya znaniy, 2006. 142 p.
19. *Maya – 3D Animation – Autodesk*. Available at: http://usa.autodesk.com/maya/ (accessed 30 March 2014).
20. Bradski G., Kaehler A. *Learning OpenCV*. USA, Sebastopol, CA, O'Reilly Media, Inc., 2008. 556 p.