УДК 004.932.2 **DOI: 10.18413/2518-1092-2016-1-3-16-23**

Рябых М.С.[1]
Сойникова Е.С.[2]
Батищев Д.С.[3]
Синюк В.Г.[4]
Михелев В.М.[5]

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ МЕТОД АНАЛИЗА И МОРФОЛОГИЧЕСКОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

1) магистрант кафедры математического и программного обеспечения информационных систем Белгородский государственный национальный исследовательский университет, ул. Победы д.85, г. Белгород, 308015, Россия. *e-mail: 828130@bsu.edu.ru*
2) инженер института высоких технологий БелГУ, 1-й Первомайский пер., 1-а, Белгород, 308001, Россия. *e-mail: 831468@bsu.edu.ru*
3) аспирант кафедры математического и программного обеспечения информационных систем. Белгородский государственный национальный исследовательский университет, ул. Победы д.85, г. Белгород, 308015, Россия. *e-mail: batishchev@bsu.edu.ru*
4) профессор кафедры программного обеспечения вычислительной техники и автоматизированных систем, кандидат технических наук, доцент, Белгородский государственный технологический университет им. В.Г. Шухова, ул. Костюкова 46, г. Белгород, 308012, Россия. *e-mail: vgsinuk@mail.ru*
5) доцент кафедры математического и программного обеспечения информационных систем, кандидат технических наук, доцент, Белгородский государственный национальный исследовательский университет, ул. Победы д.85, г. Белгород, 308015, Россия. *mikhelev@bsu.edu.ru*

Аннотация
Рассмотрена реализация алгоритма vHGW полутоновой морфологии с использованием технологий параллельного программирования OpenMP и NVIDIA CUDA. Морфологические операции имеют низкую арифметическую сложность, однако использование параллелизма по данным позволяет повысить ускорение производительности за счет применения высокоэффективных параллельных процессоров, таких как графические процессоры (GPU). Выполнение этих операций на GPU обеспечивает значительное ускорение по сравнению с реализацией на центральном процессоре для различных размеров структурирующих элементов. Показано, что алгоритм vHGW является одним из быстрых алгоритмов для вычисления дилатации и эрозии бинарных и полутоновых изображений на последовательном процессоре. Представленная реализация алгоритма vHGW для графических процессоров с использованием технологии CUDA повышает производительность морфологической обработки изображений. Показана эффективность реализации алгоритма с помощью технологии CUDA по сравнению с OpenMP при фильтрации полутонового и бинарного изображений с разным разрешением и различным размером структурирующего элемента.
Ключевые слова: математическая морфология, медицинские изображения, GPU, OpenMP, NVIDIA CUDA, vHGW, фильтрация.

UDK 004.932.2

Ryabykh M.S.[1]
Soynikova E.S.[2]
Batishchev D.S.[3]
Sinyuk V.G.[4]
Mikhelev V.M.[5]

HIGH-PERFORMANCE ANALYSIS METHOD AND MORPHOLOGICAL IMAGE PROCESSING

1) Master's Degree Student, Belgorod State National Research University, 85 Pobedy St., Belgorod, 308015, Russia *e-mail: 828130@bsu.edu.ru*
2) Engineer, Institute of High Technologies of BSU, 1-a 1st Pervomayskiy Lane, Belgorod, 308001, Russia *e-mail: 831468@bsu.edu.ru*

3) Postgraduate Student, Department of Mathematical and Software Information Systems, Belgorod State National Research University, 85 Pobedy St., Belgorod, 308015, Russia Russia
e-mail: batishchev@bsu.edu.ru
4) Candidate in Technical Sciences, Associate Professor, Department of Software Computer Technology and Automated Systems, Belgorod State Technological University named after V.G. Shoukhov, 46 Kostyukova St., Belgorod, 308012, Russia.
e-mail: vgsinuk@mail.ru
5) Candidate in Technical Sciences, Associate Professor, Department of Mathematical and Software Information Systems, Belgorod State National Research University, 85 Pobedy St., Belgorod, 308015, Russia.
e-mail: mikhelev@bsu.edu.ru

Abstract
The article discusses the implementation of the algorithm of vHGW grayscale morphology with the use of OpenMP parallel programming technology and NVIDIA CUDA. Morphological operations have low arithmetic complexity, but the use of data parallelism can improve acceleration performance with parallel processors such as graphics processors (GPU). Performing these operations for GPU provides significant acceleration compared with the central processor implementation for structuring elements of various sizes. It is shown that vHGW algorithm is a fast algorithm for computing dilation and erosion of binary and grayscale images on a serial processor. The implementation representation of vHGW algorithms for GPUs with CUDA technology improves the performance of morphological image processing. The authors demonstrate the efficiency of the algorithm implementation using CUDA technology, comparing it with the filtration OpenMP binary and grayscale images with different resolution and different size of the structuring element.
Keywords: mathematical morphology; medical imaging; GPU; OpenMP; NVIDIA CUDA; vHGW; filtration.

В связи с бурным ростом развития информационных технологий в настоящее время получает большое развитие такая область исследования, как компьютерная обработка изображений. Она используется во многих областях науки и экономики, например, в обработке медицинских изображений, распознавании текста, обработке спутниковых снимков, машинном зрении, автоматическом управлении автомобилями, определении формы интересующего объекта, определении перемещений объекта и т.д.

Операции математической морфологии являются важными строительными блоками многих алгоритмов обработки изображений, которые включают в себя удаление объектов, заполнение отверстий, сглаживание границ, скелетизацию, удаление шума и т.д.

Морфология (математическая морфология) – это инструмент для выделения и анализа на изображении графических элементов с известной геометрической структурой [2]. В общем случае морфология является ресурсоемкой операцией, поэтому на практике часто используют фильтры с прямоугольными примитивами, для которых существуют быстрые алгоритмы вычисления.

В то время как морфологические операции имеют низкую арифметическую сложность, их применимость к параллелизму работы с данными дает возможность для ускорения производительности с применением высокоэффективных параллельных процессоров. Выполнение этих операций на графических процессорах (GPU) обеспечивает значительное ускорение по сравнению с реализацией на центральном процессоре для различных размеров структурирующегося элемента. Однако, при увеличении размера такого элемента или изображения вычислительная сложность алгоритмов превышает вычислительную мощность графического процессора, и она не достигает результатов, полученных на самых быстрых алгоритмах на основе CPU.

Van Herk / Gil-Werman (vHGW) является одним из самых быстрых алгоритмов для вычисления дилатации и эрозии бинарных и полутоновых изображений на последовательном процессоре.

В данной работе представлена реализация алгоритма vHGW для графических процессоров с использованием технологии CUDA, что позволяет значительно повысить производительность по сравнению с другими алгоритмами морфологической обработки изображений. В силу высокой значимости повышения точности регистрации изображений и высокой эффективности современных методов компьютерного зрения, данная область

представляется перспективной и актуальной для современной действительности.

Для распараллеливания вычислений использованы технологии OpenMP и CUDA, так как они в большей степени обладают высокой производительностью.

Полутоновые операции дилатации и эрозии устанавливают значение каждого пикселя j в изображении до максимума и минимума (соответственно) для всех пикселей в заданной окрестности j. Операции открытия (закрытия) реализованы в виде эрозии (дилатации) с последующей дилатацией (эрозией). Окрестность, также называемая структурирующим элементом (SE), может быть произвольной формы и размера. В данной работе мы рассматриваем только прямоугольные SE.

Алгоритм vHGW (van Herk Gil-Werman), описан ван Херком [10], Гилем и Верманом [6] в своих работах.

Это алгоритм полутоновой морфологии для вычисления дилатации и эрозии со сложностью, независимой от размера структурирующего элемента. Он работает для всех структурирующих элементов, состоящих из горизонтальных и / или вертикальных линейных элементов, и требует не более 3-х сравнений значений пикселей для каждого выходного пикселя.

Для полутоновой морфологии дилатация вычисляется по формуле 1, а эрозия по формуле 2.

$$(f \oplus k)(x) = \max_{\substack{z \in K \\ (x-z) \in F}} \{f(x-z) + (z)\}\{f(x - -z) + k(z)\} \quad (1)$$

$$(f \ominus k)(x) = \min_{z \in K}\{f(x+z) - k(z)\} \quad (2)$$

Для эрозии, вместо максимального значения нужно искать минимальное.

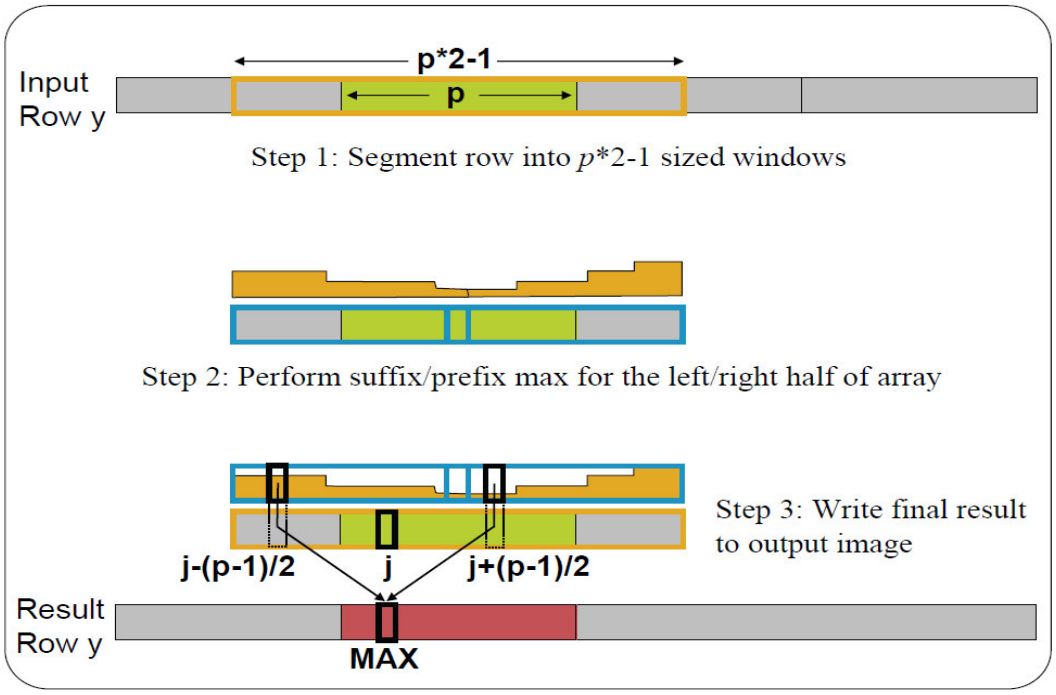

Рис. 1. vHGW алгоритм для горизонтального структурного элемента размера p
Fig. 1. vHGW algorithm for horizontal structural element of p size

Алгоритм состоит из трёх основных этапов [3, 5] (рис. 1):

I. Изображение разделяется на отрезки длины p с (p-1)/2 на каждой стороне, чтобы сформировать окно размером 2p-1, с центром в точке p-1, 2p-1, 3p-1, ...

II. Для каждого пикселя k = [0;p-1] в заданном окне w, массив префиксов R рассчитывается для пикселов слева от центра

a) $R[k] = \max(w[j]), j = k \dots (p-1)$,

а массив суффиксов S рассчитывается для пикселов справа от центра (p-1) ... (2p-2),

b) $S[k] = \max(w[p-1+j]), j = 0 \dots k$

c) R [k] и S [k] объединяются вместе, чтобы вычислить максимум фильтра

III. Для каждого пикселя (p-1)/2<=j<p+(p-1)/2 в w (отрезок длины p) результат дилатации

$$result[i] = \max(R[j-m], S[j+m]),$$
$$m = (p-1)/2$$

В результате, мы не оцениваем первые p / 2 пикселей и, в худшем случае, последние 3p / 2 пикселей. Чтобы получить разумные результаты на всех пикселях в изображении, мы вкладываем реальное изображение в более крупное изображение увеличенных размеров, где добавлены пограничные пиксели, инициализированные 0 для дилатации и 255 для эрозии.

Алгоритм vHGW выполняет дилатацию с помощью SE размером p = 2n + 1 за O (n) времени (n = число пикселей изображения). При этом выполняет менее трех сравнений на пиксель, независимо от размера SE.

Несмотря на быстроту самого алгоритма, для изображений большого размера требуется больше времени на его обработку, а если необходимо обработать коллекцию изображений, то обработка может занимать часы. Для ускорения выполнения алгоритма предлагается использовать параллельные технологии.

Исходя из приведенного выше алгоритма, можно сделать вывод, что на каждой итерации новое значение интенсивности для каждого пикселя зависит от предыдущего значения, а также от значений окружающих его пикселей.

Поэтому, чтобы иметь возможность распараллелить программу с помощью OpenMP необходимо вынести некоторые вычисления в отдельные функции. После этого можно вставить директиву *#pragmaompparallelforprivate(prefix, suffix)* перед основным циклом алгоритма.

Технология OpenMP использует для распараллеливания модель «ветвление-слияние» (fork-joinparallelism), показанную на рисунке 2. Программа начинает выполняться в главном потоке, и когда встречается параллельная секция, создаются новые потоки, и нагрузка распределяется уже по нескольким потокам, после того как все потоки выполнились, снова начинает работать только один главный поток. С точки зрения программирования – это очень удобный подход, т.к. необходимо в последовательную часть программы вставить всего лишь одну или несколько директив [1].

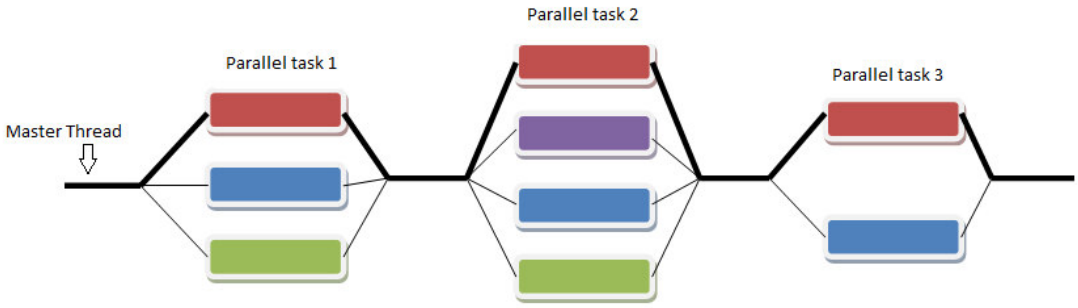

Рис. 2. Модель «ветвление-слияние»
Fig. 2. Model "fork-joint parallelism"

Представленная реализация vHGW на CUDA использует следующие детали:

• Отдельный код для горизонтальной и вертикальной версии структурирующего элемента.

• Один поток вычисляет максимальное (минимальное) значения сразу для двух массивов S [K], R [K]

• Разделяемая память не используется, поскольку это ограничивает использование мультипроцессоров CUDA с увеличением размера изображений и структурирующих элементов.

При этом необходимо учесть следующие определения CUDA:

• CUDA мультипроцессор создаёт, управляет, планирует и выполняет потоки в группах из 32 параллельных нитей, называемых warp'ами [9]

• CUDA мультипроцессор содержит ряд блоков нитей, с максимумом 8 блоков нитей в CUDA CC версии 2.x и ниже.

• Размер блока должен быть кратным 32 нитям (warp) с рекомендуемым минимумом 64 (2 warp'а).

Если предположить, что возможно использовать только 32 потока (один warp), мы можем выполнить горизонтальную дилатацию на изображении с 1100 столбцов в строке и структурного элемента размером 11, с

использованием более 32KB разделяемой памяти следующим образом:

• структурный элемент представляет собой горизонтальную линию 11 пикселов, поэтому p=11;

• каждая строка изображения составляет 1100 пикселей, поэтому 1100 / p = 100 окон;

• 2 массива в одном окне с помощью 2p значений и 2 нитей;

• каждый блок использует 32 потока;

• поэтому каждый блок требует 100*2p*32/2≈34K разделяемой памяти, которая равна 34KB для 8 битных целочисленных изображений или 136 КБ для 32 битных изображений с плавающей точкой;

• каждый мультипроцессор имеет 48kb разделяемой памяти для карт, поддерживающих CUDA CC 2,0 или выше [9].

71% разделяемой памяти мультипроцессора расходуется на изображение с 8 битами точности (для структурирующего элемента, размером 11 пикселей), а также превышает её в 1,8 раза для 32 битного изображения с плавающей точкой.

Распределение ресурсов, которое выделяет все 48kb разделяемой памяти на один блок с 32 нитями в блоке только 1 активный warp для этого мультипроцессора. Только с одним активным warp'ом из возможных 48 (CUDACC 2.0) [7] заполняемость (коэффициент использования нити) всего 2% от возможностей мультипроцессора, что значительно ограничивает потенциал производительности.

С учетом заявленной цели анализа и обработки больших изображений размером более одного миллиона пикселей, с 8 и более бит точности, максимум разделяемой памяти для массивов vHGW не является достаточным.

Для тестирования алгоритма морфологической обработки изображении vHGW, реализованного с помощью технологий OpenMP и CUDA, исходное изображение, представленное на рисунке 3, подавалось на вход 100 раз для более точного измерения времени его выполнения. Каждый проделанный эксперимент проводился с одним изображением, но разного разрешения. После выполнения обработки изображений, программа производила подсчет времени выполнения операций над изображением.

Рис. 3. Исходное изображение земной поверхности
Fig. 3. The original image of the Earth's surface

На рисунке 3 представлено исходное полутоновое изображение земной поверхности, снятое из космоса, а на рисунке 4 – результат эрозии этого изображения структурирующим элементом размера 3×3. Заметно, что изображение стало немного темнее, что вполне логично, учитывая то, что алгоритм находит локальный минимум интенсивности в заданной окрестности.

Рис. 4. Результат эрозии исходного изображения
Fig. 4. The result of erosion of the original image

Одной из главных проблем обработки изображений является присутствие шума в изображении. Эту проблему можно решить с помощью морфологической фильтрации [4, 8].

На рисунке 5 представлено исходное бинарное зашумленное изображение отпечатка пальца. С помощью последовательности операций дилатации и эрозии удалось избавиться от шума, что продемонстрировано на рисунке 6.

Рис. 5. Исходное зашумленное изображение отпечатка пальца
Fig. 5. Original noisy fingerprint image

Рис. 6. Результат морфологической фильтрации изображения отпечатка пальца
Fig. 6. The result of morphological filtering of a fingerprint image

Как видно из графика, представленного рисунке 7, при увеличении размера структурного элемента, время выполнения алгоритма не только не увеличивается, а даже уменьшается, как с использованием 1 потока, так и 12-ти потоков.

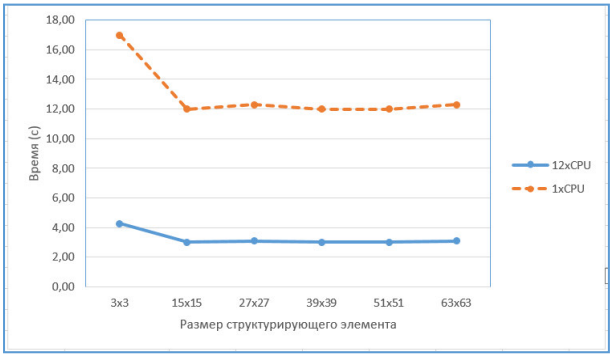

Рис. 7. Зависимость времени выполнения от размера структурирующего элемента для программы на 1 и 12 потоках
Fig. 7. Dependence of the execution time of the size of a structuring element for the application on 1 and 12 threads

На рисунке 8 видим, что с увеличением разрешения изображения программа с одним потоком работает гораздо медленнее, чем с 12-ю потоками.

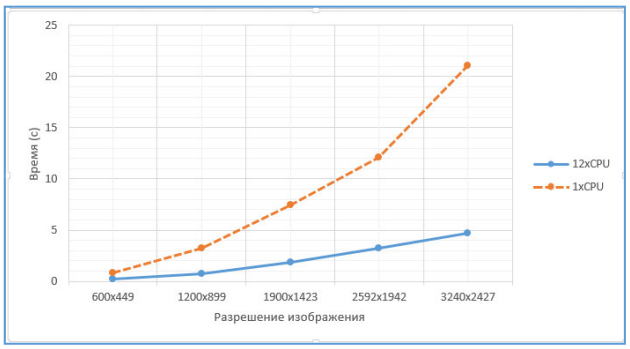

Рис. 8. Зависимость времени выполнения от разрешения изображения для программы на 1 и 12 потоках
Fig. 8. Dependence of the execution time of the image resolution for the program on 1 and 12 threads

Таким образом, анализируя представленные результаты, можно сделать вывод, что реализованные алгоритмы на CPU требуют большого количества времени выполнения. Этот факт свидетельствует о том, что необходимо улучшить эффективность алгоритмов для уменьшения времени выполнения программы.

Для того, чтобы улучшить эффективность алгоритма морфологической обработки изображений, данный алгоритм был реализован на GPUс помощью технологии NVIDIACUDA.

На рисунке 9. приводится график зависимости времени выполнения алгоритма от размера структурирующего элемента в трёх случаях:

- на 12 потоках CPU;
- с использованием стандартных морфологических операций на CUDA библиотеки OpenCV;
- с использованием алгоритма, рассматриваемого в данной работе на GPU.

Как видно из графика, представленного рисунке 9, алгоритм vHGW и его реализация на GPU гораздо эффективнее, чем алгоритмы, использующиеся в OpenCV.

Таким образом, можно сделать вывод, реализация алгоритма vHGW с использованием технологии NVIDIA CUDA позволяет получить ускорение приблизительно в 15 раз большее, по сравнению с использованием технологии OpenMP.

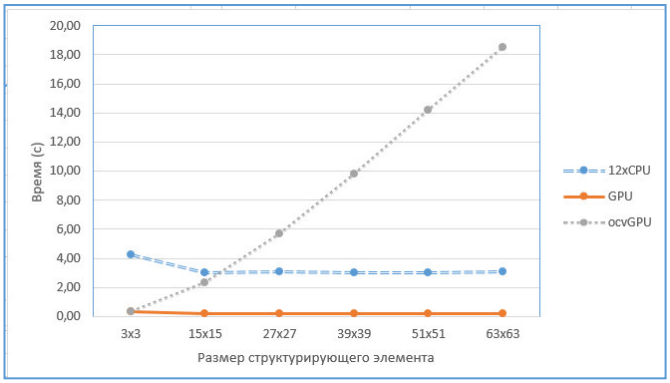

Рис. 9. Зависимость времени выполнения от размера структурирующего элемента
Fig. 9. Dependence of the execution time of the size of the structuring element

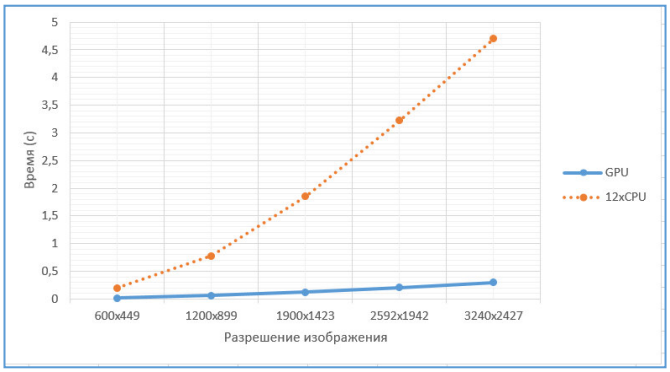

Рис. 10. Зависимость времени выполнения от размера изображения
Fig. 10. Dependence of the execution time of the image size

По графикам зависимости времени выполнения программы от разрешения исходного изображения, представленных на рисунке 10, также можно сделать вывод, что для сложных операция над пикселами изображения эффективнее всего использовать технологию NVIDIA CUDA.

В процессе выполнения данной работы рассмотрена реализация алгоритма vHGW полутоновой морфологии с использованием технологий параллельного программирования OpenMP и NVIDIA CUDA. Показано, что реализация алгоритма vHGW для графических процессоров с использованием технологии CUDA повышает производительность морфологической обработки изображений. Показана эффективность реализации алгоритма с помощью технологии CUDA по сравнению с OpenMP при фильтрации полутонового и бинарного изображений с разным разрешением и различным размером структурирующего элемента.

Список литературы

1. Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие. – М.: Изд-во МГУ, 2009. – 77 с.

2. Гонсалес Р., Вудс Р. Цифровая обработка изображений [Текст]. / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с.

3. Батищев Д.С., Михелев В.М. Инфраструктура высокопроизводительной компьютерной системы для реализации облачных сервисов хранения и анализа данных персональной медицины. Научные ведомости Белгородского государственного университета. Серия: Экономика. Информатика. 2016. Т. 37. № 2 (223). С. 88-92.

4. Рябых М. С., Сойникова Е. С., Батищев Д.С., Михелёв В.М. Морфологическая обработка изображений отпечатков пальцев с использованием параллельных вычислений на графических процессорах // «Тенденции развития науки и образования: сборник научных трудов, по материалам XII международной научно-практической конференции» (31 марта 2016 г.) Часть 4. – Самара: Издательство НИЦ «Л-Журнал», 2016. – 60 с.

5. Domanski L., Vallotton P., Wang D., “Parallel vHGW image morphology on CPUs using CUDA”, CSIRO, Mathematical and Informational Sciences, Biotech Imaging.

6. Gil J. and Werman M.: Computing 2-D Min, Median, and Max Filters, IEEE Trans. Pattern Anal. Mach. Intel., 1993, Vol 15, Number 5, 504–507.

7. Kirk D. B. and Hwu W. mei W., Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan Kaufmann, 2010.

8. Soinikova E.S., Ryabihk M.S., Batishchev D.S., Mikhelev V.M. High-performance method for boundary detection in medical images// Academic science – problems and achievements IX: Proceedings of the Conference. North Charleston, 20-21.06.2016—North Charleston, SC, USA:CreateSpace, 2016, p.93-95.

9. NVIDIA, “NVIDIA CUDA C programming guide – version 7.0,” NVIDIA developer website, June 2016. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz4IHtkC9CZ.

10. Van Droogenbroeck M., “On the Implementation of Morphological Operations”, Math. Morphology and its applications to image processing, J. Serra and P. Sollie, eds. Dordrecht: Kluwer Academic Publishers, 1994, pp. 241-248.

References

1. Antonov A.S. Parallel Programming with OpenMP Technology. Moscow: MGU, 2009. 77 p.

2. Gonzalez R., Woods R. Digital Image Processing. Moscow: Tekhnosfera, 2005. 1072 p.

3. Batishchev D.S, Mikhelev V.M The Infrastructure of High-performance Computer System for the Implementation of Cloud Storage and Analysis of Personal

Medical Data. Scientific Bulletin of Belgorod State University. Series: Economy. Computer Science. 2016. Vol. 37. № 2 (223). Pp. 88-92.

4. Ryabykh M.S., Soynikova E.S., Batishchev D.S., Mikhelev V.M. Morphological Processing of Fingerprint Images with the Use of Parallel computing on GPUs // "Trends in the Development of Science and Education: a Collection of Scientific Papers, based on the materials of the XII International Scientific-Practical Conference". Part 4. Samara: NITS «L-Zhurnal», 2016. 60 p.

5. Domanski L., Vallotton P., Wang D., "Parallel vHGW image morphology on CPUs using CUDA", CSIRO, Mathematical and Informational Sciences, Biotech Imaging.

6. Gil J. and Werman M.: Computing 2-D Min, Median, and Max Filters, IEEE Trans. Pattern Anal. Mach. Intell., 1993, Vol 15, Number 5, 504–507.

7. Kirk D. B. and Hwu W. mei W., Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series). Morgan Kaufmann, 2010.

8. Soinikova E.S., Ryabihk M.S., Batishchev D.S., Mikhelev V.M. High-performance method for boundary detection in medical images// Academic science – problems and achievements IX: Proceedings of the Conference. North Charleston, 20-21.06.2016—North Charleston, SC, USA:CreateSpace, 2016, p.93-95.

9. NVIDIA, "NVIDIA CUDA C programming guide – version 7.0," NVIDIA developer website, June 2016. [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz4IHtkC9CZ.

10. Van Droogenbroeck M., "On the Implementation of Morphological Operations", Math. Morphology and its applications to image processing, J. Serra and P. Sollie, eds. Dordrecht: Kluwer Academic Publishers, 1994, pp. 241-248.