REFERENCES

1. Zykov A.A. *Osnovy teorii grafov* [Graph theory bases]. Moscow, Vuzovskaya kniga, 2004. 664 p.
2. Demer M., Emmert-Shtrayb F., Tsoy U.P., Varmuza K. Noviy funktsional informativnosti dlya analiza struktury khimicheskikh grafov [A new information content functional to analyze the structure of kinetic graphs]. *Bulletin of the Tomsk Polytechnic University*, 2010, vol. 316, no 5, pp. 5–11.
3. Pogrebnoy An.V., Pogrebnoy V.K., Invariant grafa na osnove kompaktnykh podgrafov i algoritm ego vychisleniya [Graph invariant on the basis of compact subgraph and algorithm for its computation]. *Bulletin of the Tomsk Polytechnic University*, 2013, vol. 322, no. 5, pp. 200–204.
4. Pogrebnoy V.K. Metod integratsii strukturnikh razlichii v grafovykh modelyakh i ego primenenie dlya opisaniya struktur [Methods of integration of structural differences in graph models t odescribe structures]. *Bulletin of the Tomsk Polytechnic University*, 2011, vol. 318, no. 5, pp. 10–16.
5. Pogrebnoy V.K. Zadacha opredeleniya otsenok skhodstva struktur dvukh grafov na osnove vydeleniya obshchikh chastei [The task of estimating the similarity of two graphs structures on the basis of general parts definition]. *Bulletin of the Tomsk Polytechnic University*, 2013, vol. 322, no 5, pp. 194–199.
6. Pogrebnoy V.K., Pogrebnoy An.V. Issledovanie polinomialnosti metoda vychisleniya integralnogo opisatelya strukturi grafa [Studying the polynomiality of computation technique for integrated descriptor of graph structure]. *Bulletin of the Tomsk Polytechnic University*, 2013, vol. 323, no. 5, pp. xx–xx.

УДК 004.032.24

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА ВЫДЕЛЕНИЯ ГРАНИЦ ОБЪЕКТОВ НА ОСНОВЕ СТРУКТУРНО-ГРАФИЧЕСКОГО ПРЕДСТАВЛЕНИЯ

А.Ю. Демин, В.А. Дорофеев

Томский политехнический университет
E-mail: ad@tpu.ru

Актуальность работы *обусловлена необходимостью исследования возможности распараллеливания алгоритмов на основе структурно-графического представления.*
Цель работы*: Распараллелить алгоритм нахождения границ на аэрофотоснимках с помощью структурно-графической формы.*
Методы исследования: *Обработка растровых изображений в интеллектуальных системах навигации и управления с помощью линейных фильтров. Представление алгоритмов в графовой форме: дерева операторов, блок-схемы, графа потока данных. Проектирование программных средств с помощью средств платформы .Net библиотеки Task Parallel Library.*
Результаты: *В работе рассматривается программная реализация алгоритма нахождения границ объектов на изображениях с помощью оператора Собеля. Программная реализация представлена в структурно-графической форме. Предложен полуавтоматический способ распараллеливания рассматриваемой программной нагрузки. Программно реализован распараллеленный алгоритм, и проведен анализ эффективности распараллеливания для различных изображений.*

Ключевые слова:
Оператор Собеля, обработка изображений, дерево операторов, граф потока данных, распараллеливание программ.

Введение

Интеллектуальные системы навигации и управления (ИСНУ) должны обладать современными средствами связи и мониторинга, обеспечивающими сбор, накопление и обработку информации, поступающей с подвижных и стационарных, в том числе труднодоступных, объектов, выработку решений и оперативную доставку команд управления на все уровни и на все объекты, осуществлять автоматизированное документирование информационных обменов, результатов обработки информации и принятых решений, обеспечивать непрерывный мониторинг состояния объектов, отображение местоположения и маршрутов движения мобильных групп и подвижных объектов на электронных картах диспетчерских пунктов в реальном масштабе времени и целый ряд других функций [1].

Использование спутниковых и аэрофотоснимков, изображений, получаемых с беспилотных летательных аппаратов, позволило в телекоммуникационных технологиях существенно улучшить возможности построения аппаратно-программных комплексов для сопровождения, мониторинга и управления мобильными объектами [2].

Важной задачей при анализе изображений, полученных с различных подвижных и труднодоступных объектов, в том числе со спутников, является проблема распознавания различных объектов, таких как различные водоемы (реки, озера, водохранилища, затопленные территории), площади выгоревших лесов, промышленные и гражданские объекты и т. д. Решение подобных задач актуально при построении ИСНУ для Авиалесоохраны РФ, Россельхознадзора РФ, Росгидромета РФ и гидрометов стран СНГ, различных служб МЧС и силовых ведомств.

С другой стороны, изображения, получаемые со спутников и других объектов, характеризуются большим объемом данных. Так, файл с одним

спутниковым снимком может иметь размер в несколько гигабайт. Этот факт показывает необходимость разработки эффективных методов обработки таких изображений с привлечением высокопроизводительных и многоядерных аппаратных средств, в том числе суперкомпьютеров.

Нахождение границ с помощью разностных фильтров

Одним из этапов распознавания образов в системах компьютерного зрения может выступать алгоритм нахождения границ с помощью разностных фильтров. При этом используют семейство операторов пространственного дифференцирования, применяемых для вычисления приближенных значений градиента яркости изображения. Результатом применения такого оператора в каждой точке изображения является либо вектор градиента яркости в этой точке, либо его норма [3].

В каждом пикселе градиент показывает направление наибольшего увеличения яркости. Длина вектора градиента определяет величину изменения яркости, а значит, вероятность нахождения точки на границе.

Одним из способов нахождения границ на изображении является реализация фильтра или оператора Собеля (Sobel) [3], который позволяет найти неточное приближение градиента яркости изображения. Достоинством оператора Собеля является наилучшая реакция на ступенчатый перепад и наименьший коэффициент утолщения контурной линии [4].

Формально оператор Собеля определяется следующим образом:

Пусть A – исходное изображение, а G_x и G_y – два изображения, где каждая точка содержит приближенные производные по x и по y. Они вычисляются следующим образом [3]:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A, \quad G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \quad (1)$$

где * обозначает двухмерную операцию свертки. Двухмерная операция свертки проводится по формуле:

$$B(x, y) = \sum_i \sum_j F(i, j) \cdot A(x + i, y + j), \quad (2)$$

где $F(i,j)$ – ядро фильтра (в случае оператора Собеля это матрицы 3×3, представленные выше (1)).

Величину градиента в каждом пикселе изображения можно вычислить следующим образом.

$$G = \sqrt{G_x^2 + G_y^2}. \quad (3)$$

Направление градиента определяется так:

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right).$$

Операции свертки G_x и G_y можно использовать отдельно для нахождения вертикальных и горизонтальных границ.

Структурно-графическое представление программной реализации оператора Собеля

Исходные данные для программной реализации представляют собой массив байт – значений яркостей пикселей (в случае полутонового изображения), либо значений яркостей отдельных каналов в случае использовании различных цветовых моделей. В случае использования RGB модели используется классическое расположение данных для формата bmp в памяти попиксельно (B1 G1 R1 B2 G2 R2 ...).

Программная реализация оператора Собеля для одноядерного процессора является тривиальной задачей и на псевдокоде выглядит следующим образом:

```
for all y
        for all x
                Sobel compute
```

Здесь внешний цикл организует перебор всех строк исходного изображения, а внутренний цикл – перебор всех пикселей в строке. Тело циклов Sobel compute вычисляет значение по Собелю для одного пикселя и на псевдокоде представляется также двумя вложенными циклами:

```
for all i
        for all j
        {
                Gx compute
                Gy compute
                G compute
        }
```

Здесь оба цикла организуют перебор в окрестности точки и вычисления двухмерной операции свертки (2). В теле циклов вычисляются приближенные производные по x и по y (1) и величина G (3).

Для анализа и представления программной реализации используем способ, описанный в [5]. Данный способ позволяет анализировать текст программы и представить программную нагрузку в виде взаимосвязанных деревьев (дерева операторов, дерева данных, дерева функций и т. д.), графа управляющих связей, графа потока данных.

Вычисление значений по Собелю для одного пикселя представим в виде дерева операторов, представленного на рис. 1.

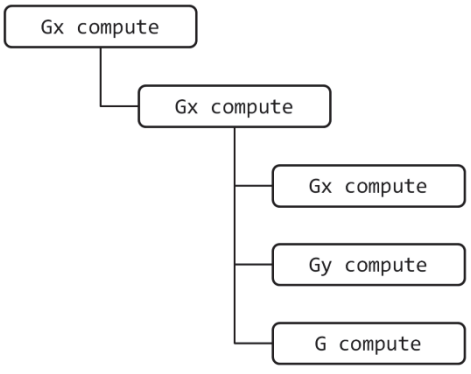

Рис. 1. *Дерево операторов для вычисления цвета одного пикселя*

Исходя из представления в виде дерева по алгоритму, изложенному в [5], получим блок-схему (рис. 2).

Определение 1. *Блок-схема* – это ориентированный граф $H=(NU,n_0)$ с вершинами специального вида, соответствующий условиям:

1. Граф G имеет единственную входную вершину n_0, и в эту вершину не входит ни одна дуга.
2. Граф G не содержит параллельных дуг и петель.
3. Конечное множество дуг $U=\{u_f\}$, $f=1,\ldots,F$, отражает отношение предшествования по выполнению между операторами программы.
4. Конечное множество вершин $N=\{n_\mu\}$, $\mu=0,\ldots,M$, представляет совокупность операторов $B=\{b_\mu\}$, $\mu=0,\ldots,M$, программы. Между ними существует взаимно однозначное соответствие, (каждая вершина представляет только один оператор программы и, наоборот, каждый оператор программы в графе представляется только одной вершиной).
5. Конечное множество вершин $N=\{n_\mu\}$, $\mu=0,\ldots,M$, представляет совокупность сложно-составных операторов $B'=\{b'_\mu\}$, $\mu=0,\ldots,M$, программы. Между ними существует взаимно однозначное соответствие. Поскольку любой вершине n_μ соответствует сложно-составной оператор, то она представляет группу простых операторов.

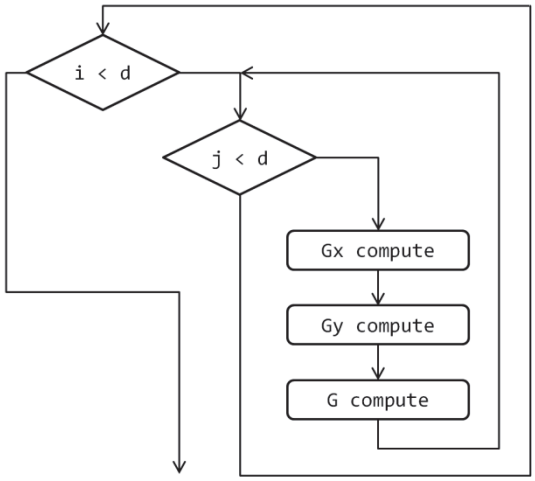

Рис. 2. *Блок-схема функции для вычисления цвета одного пикселя*

По алгоритму, представленному в [5], получим граф потока данных (ГПД) для программной функции вычисления цвета пикселя по Собелю (рис. 3).

Определение 2. *Графом потока данных* (ГПД) называется ориентированный граф $R=(N,D,U)$, где N – множество вершин, соответствующих сложно-составным операторам, представленных листьями в динамически строящемся дереве; D – множество вершин, соответствующих данным, используемым в программе, причем единице данных могут быть поставлены в соответствие несколько вершин; U – множество таких дуг, что $\exists(d_i,n_j)\in U$, если $d_i\in D$ используется в $n_j\in N$, либо если $d_i\in D$ изменяется в $n_j\in N$.

Определение 3. Множество вершин N из R, соответствующих сложно-составным операторам, назовем *переходами*, а множество вершин D из R, соответствующих данным, назовем *позициями*.

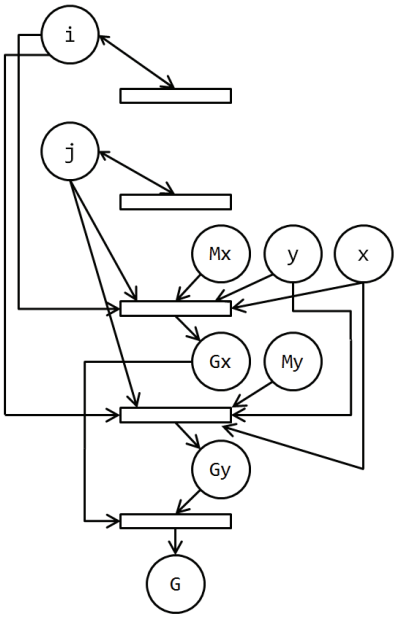

Рис. 3. *ГПД для вычисления цвета одного пикселя*

Определение 4. Подмножества Ω_i множества O называются *ярусами*. Множество операторов Ω_1 первого яруса тождественно множеству операторов, которые информационно независимы. Множество операторов Ω_2 второго яруса тождественно множеству операторов, информационно зависимых хотя бы от одного оператора первого яруса. Множество операторов Ω_i i-го яруса тождественно множеству операторов, информационно зависимых хотя бы от одного оператора $(i-1)$-го яруса. Также для любого яруса выполняются условия $Zi\Omega_i\neq\varnothing\Omega_i\cap\Omega_j=\varnothing(i\neq j)$

Определение 5. ГПД, соответствующий упорядочиванию функциональных операторов по ярусам, отвечающим подмножествам $\Omega_1,\Omega_2,\ldots,\Omega_i,\ldots$ множества O, называется ГПД *в ярусно-параллельной форме* (ЯПФ).

Для анализа возможности распараллеливания перестроим ГПД в ЯПФ (рис. 4.) по следующему алгоритму.

Алгоритм 1. Алгоритм перестроения ГПД в ЯПФ

Шаг 1. Упорядочить множество переходов N так, чтобы каждый переход занимал свой отдельный ярус с 0 по N. И чтобы их порядок не противоречил порядку выполнения соответствующих операторов. Установить $i=1$.

Шаг 2. Пронумеровать все переходы в соответствии с занимаемыми ярусами.

Шаг 3. Если для перехода n_i выполняется условие $\Gamma^+(n_i)\cap\Gamma^-(n_j)\neq\varnothing$ для любого n_j, принадлежащего предшествующему ярусу, то перейти к шагу 9.

Шаг 4. Если для перехода n_i выполняется условие $\Gamma^+(n_i)\cap\Gamma^+(n_j)\neq\varnothing$ для любого n_j, принадлежащего предшествующему ярусу, то перейти к шагу 9.

Шаг 5. Если для перехода n_i выполняется условие $\Gamma^-(n_i)\cap\Gamma^+(n_j)\neq\varnothing$ для любого n_j, принадлежащего предыдущему ярусу, то перейти к шагу 9.

Шаг 6. Исключить переход n_i с яруса, к которому он принадлежал $\Omega_k=\Omega_k\backslash n_i$. Включить переход n_i на предыдущий ярус $\Omega_{k-1}=\Omega_{k-1}\cup n_i$.

Шаг 7. Если ярус, на котором находился переход n_i, пуст ($\Omega_k=\varnothing$), то удалить ярус Ω_k из списка ярусов.

Шаг 8. Перейти к шагу 3.

Шаг 9. Увеличить i ($i=i+1$). Если $i>|N|$, то перейти к шагу 10, иначе перейти к шагу 3.

Шаг 10. Конец.

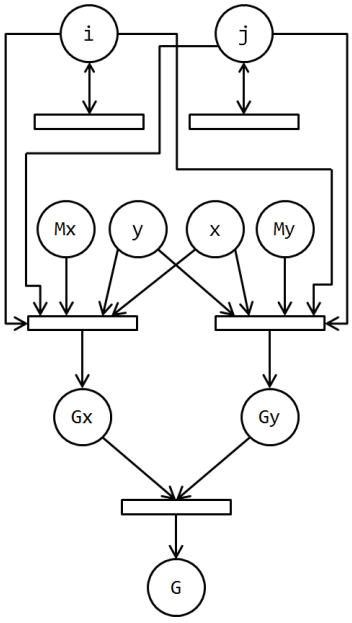

Рис. 4. *ГПД в ЯПФ для вычисления градиента в указанной точке*

Представление программы в виде ГПД в ЯПФ показывает отсутствие обратных внутренних связей по данным в теле циклов, что допускает принципиальную возможность параллельного выполнения этих вычислений. Максимальное число процессоров в этом случае равно девяти. Однако большие временные затраты на организацию запуска вычислительных процессов на этом уровне делают нецелесообразным распараллеливание этой подпрограммы.

Исследуем подпрограмму вычисления оператора Собеля на всем изображении. Данные вычисления, как показано выше, организуются также через два вложенных цикла по x и по y. Исследовав данный код с помощью представления деревом операторов, блок-схемы, ГПД в ЯПФ, получим аналогичные результаты возможности распараллеливания путем разбиения на n вычислительных процессов. Каждый вычислительный процесс может выполняться одновременно с другими на своем ядре и обрабатывать участок изображения из k строк. Таким образом, изображение, состоящее из y строк, разбивается на n горизонтальных областей из k строк. Каждая область обрабатывается параллельно.

Программная реализация

Для программной реализации алгоритма Собеля был выбран язык C# и библиотека Task Parallel Library (TPL), которая входит в состав платформы .NET начиная с версии 4.0.

Поскольку данная задача реализует достаточно простой алгоритм, но при этом работает с большим изображением, то логичным выбором было осуществить распараллеливание программы с помощью декомпозиции по данным. Однако стандартные классы .NET не поддерживают общий доступ из нескольких потоков, поэтому извлечение данных из изображения для потоков осуществлялось заранее с помощью стандартных средств классов .NET для работы с внутренним представлением графических данных (операции LockBits класса Bitmap). Таким образом, потоки обращались уже к простому массиву с компонентами цветового изображения, представленными байтами R, G и B. Это позволило избежать существенного снижения производительности, которое неизбежно возникает при использовании стандартных методов обращения к пикселям изображения.

Поскольку каждый пиксель представлен тремя байтами, не предпринималось дополнительных усилий по оптимальному размещению данных в памяти. Результирующее изображение также формировалось в виде битового массива в памяти, а по окончании вычислений преобразовывалось в изображение. Время, затрачиваемое на эти дополнительные операции, не включалось в общее время расчётов.

Для декомпозиции по данным общее количество строк изображения разбивалось на требуемое количество потоков, и каждый поток получал свои граничные значения, исходя из которых и производил обработку. Небольшой особенностью распределения стал размер массива с коэффициентами: поскольку в общем случае линейной фильтрации возможно использование не только ядра 3×3, но и ядер размерностью 5×5, 7×7 и т. д., необходимо корректировать и отступ первого и последнего потоков от края изображения. Новый поток создаётся, получает границы своей области и сразу запускается.

В результате выполнения программы из исходного изображения со спутникового снимка было получено изображение с выделенными объектами. На завершающем этапе обработки изображения использовалась пороговая фильтрация (рис. 5).

Анализ эффективности параллельного выполнения алгоритма определения границ

Для оценки эффективности распараллеливания данной задачи были выполнены тестовые запуски с разным количеством потоков. Запуски производились на типовой рабочей станции, оснащённой процессором второго поколения Intel Core i7–2600 с 8 Гб оперативной памяти. Чтобы минимизировать вклад сторонних факторов в общие результаты, каждый тест выполнялся 10 раз, полу-

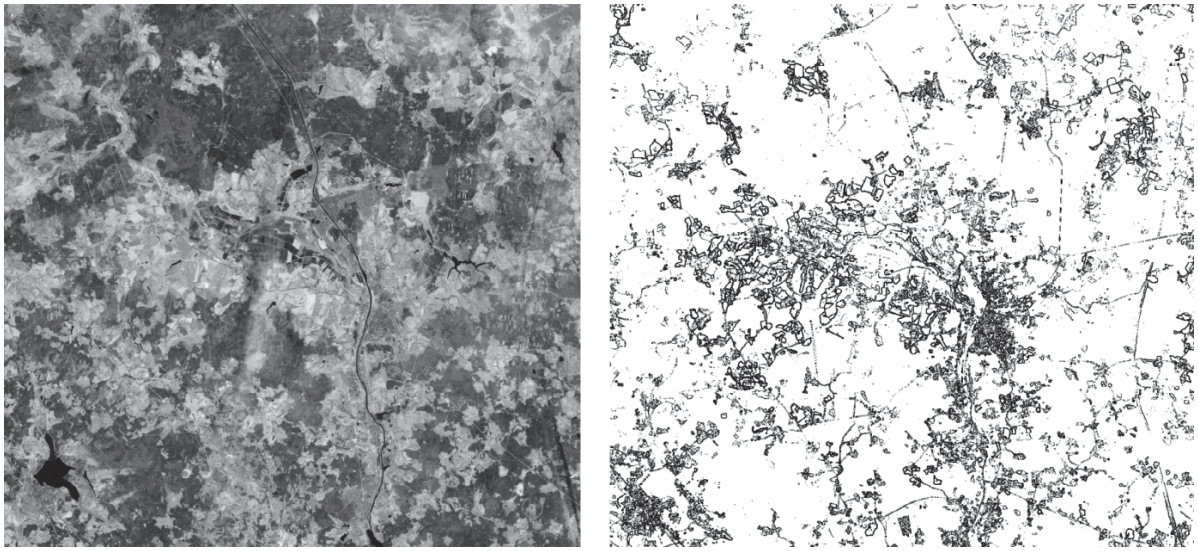

Рис. 5. *Исходное изображение и полученный результат*

ченное время усреднялось. Полученные результаты представлены на рис. 6.

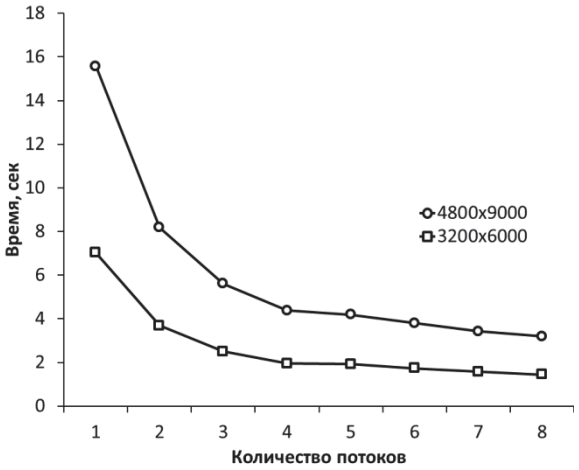

Рис. 6. *Графики зависимости влияния многопоточности на обработку изображения*

Как видно из графиков, максимальный прирост производительности обеспечивается при переходе от однопоточности к двум потокам: время обработки сокращается примерно в два раза. Каждый последующий поток также увеличивает производительность, но уже не так сильно: вероятно, сказываются аппаратные блокировки потоков при обращении к общей памяти. Из-за большого объема исходного изображения принцип локальности не работает и ядрам процессора приходится часто копировать новые фрагменты данных в кэш.

Стоит отметить, что при задействовании дополнительных ядер наблюдается более существенный прирост производительности при работе с изображениями большего размера. Это вызвано дополнительными «накладными расходами» на организацию работы самих потоков.

Поскольку для тестирования использовался четырёхядерный процессор, то начиная с пяти потоков выигрыш в производительности становится крайне незначительным (однако по-прежнему есть благодаря технологии гиперпоточности, присутствующей в процессорах компании Intel).

Заключение

1. Исследован алгоритм вычисления оператора Собеля. Программа, реализующая алгоритм, представлена в структурно-графической форме: в виде дерева операторов, блок-схем вычислительных функций, графа потока данных.
2. Предложен алгоритм перестроения графа потока данных в ярусно-параллельную форму, что позволило исследовать возможность распараллеливания программного кода. Таким образом, реализован полуавтоматический метод распараллеливания программ.
3. С помощью библиотеки Task Parallel Library программно реализована распараллеленная версия алгоритма, реализующая нахождение границ объектов на больших аэрофотоснимках.
4. Проведен анализ эффективности параллельного выполнения алгоритма определения границ. Показано существенное уменьшение времени выполнения программы.

Работа выполнена в рамках госзадания «НАУКА», регистрационный номер НИР: 4.318.2012

СПИСОК ЛИТЕРАТУРЫ

1. Сонькин М.А. Ямпольский В.З. Интегрированные системы мониторинга для труднодоступных и подвижных объектов. – Томск: Изд-во НТЛ, 2010 г. – 123 с.
2. Сонькин М.А., Ямпольский В.З. Навигационно-телекоммуникационные системы мониторинга подвижных объектов, мобильных групп и центров управления // Проблемы информатики. – 2011. – Вып. 2 (10). – С. 4–10.
3. Оператор Собеля // Википедия. 2013–2013. Дата обновления: 13.03.2013. URL: http://ru.wikipedia.org/?oldid=53441014 (дата обращения: 13.03.2013).
4. Ватутин Э.И., Мирошниченко С.Ю., Титов В.С. Программная оптимизация оператора Собела с использованием SIMD-расширений процессоров семейства x86 // Телекоммуникации. – 2006. – Вып. 6. – С. 12–16.
5. Демин А.Ю. Рейзлин В.И. Анализ программного обеспечения на основе структурно-графического представления // Проблемы информатики. – 2011. – Специальный выпуск. – С. 6–15.

Поступила 29.04.2013 г.

UDC 004.032.24

PARALLELIZATION OF ALGORITHM FOR DETECTING BORDERS ON THE BASIS OF STRUCTURAL AND GRAPHIC PRESENTATION

A.Yu. Demin, V.A. Dorofeev

Tomsk Polytechnic University

The urgency *of the discussed issue is caused by the need to investigate the possibility of parallel algorithms based on the structural and graphic presentation.*

The main aim of the study *is to parallelize the algorithm for finding the boundaries on aerial photographs using structural and graphic form.*

The methods used in the study *are the raster image processing in intelligent navigation and control using linear filters; presenting of algorithms in graph form: the operator tree, flowcharts, data flow graph; designing software by means of the platform .Net and Task Parallel Library.*

The results: *The paper describes the software implementation of the algorithm for finding the boundaries of objects in images using Sobel operator. Software implementation is presented in the structural and graphic form. The authors propose a semi-automatic approach to parallelization of the considered software load. A software algorithm is parallelized and the effectiveness of parallelization for different images is analysis.*

Key words:
Sobel operator, image processing, tree of operators, data flow graph, parallel programming model.

REFERENCES

1. Sonkin M.A. Yampolskiy V.Z. *Integrirovannye sistemy monitoring dlya* trudnodostupnykh *i podvizhnykh obektov* [Integrated monitoring system for remote and mobile objects]. Tomsk, NTL Publ., 2010. 123 p.
2. Sonkin M.A., Yampolskiy V.Z. Navigatsionno-telekommunikatsionnye sistemy monitoringa *podvizhnykh* obektov, mobilnykh grupp i tsentrov upravleniya [Navigation and communication systems for moving objects monitoring, mobile units and control centers]. *Problemy informatiki*, 2011, no. 2 (10), pp. 4–10.
3. *Operator Sobelya. Wikipedia.* 2013–2013. Retrieved: 13.03.2013. Available at: http://ru.wikipedia.org/? oldid=53441014 (accessed 13 March 2013).
4. Vatutin E.I., Miroshnichenko S.Yu., Titov V.S. Programmnaya optimizatsiya operatora *Sobela* s ispolzovaniem SIMD-rasshireniy protsessorov semeystva x86 [Software optimization of Sobel operator using SIMD-extensions of the x86 family processors]. *Telekommunikatsii*, 2006, no. 6, pp. 12–16.
5. Demin A.Yu. Reyzlin V.I. Analiz programmnogo obespecheniya na osnove strukturno-graficheskogo predstavleniya [Analysis of software based on structural graphical representation]. *Problemy informatiki*, 2011, Special Issue, pp. 6–15.