ИНФОРМАТИКА

2014 октябрь-декабрь № 4

УДК 004

Э.Н. Середин, Б.А. Залесский

АЛГОРИТМ ИНТЕРАКТИВНОГО ВЫДЕЛЕНИЯ ЛИНЕЙНЫХ ОБЪЕКТОВ НА АЭРОФОТОСНИМКАХ И КОСМИЧЕСКИХ ИЗОБРАЖЕНИЯХ

Рассматривается новый подход к решению задачи выделения линейных объектов на аэрофотоснимках и космических изображениях. Предлагается алгоритм интерактивного выделения линейных объектов с помощью ломаных линий, разработанный на основе описанного подхода. Преимуществами алгоритма являются возможность его быстрого выполнения, легкая настройка для практической работы, возможность удобной ручной корректировки решения. Алгоритм обеспечивает высокую точность выделения объектов на космических снимках среднего и низкого разрешения. Описываются особенности последовательных и параллельных вычислительных и программных реализаций алгоритма. Разработанные подходы к реализациям предложенного алгоритма позволили уменьшить на несколько порядков объем требуемых вычислений, а также сократить время его выполнения практически до долей секунды. Приводятся оценки быстродействия версий алгоритма, оптимизированных для выполнения на видеокарте.

Введение

В последние годы все большую значимость приобретает актуальная картографическая информация. Карты местности повсеместно используются в строительстве, навигации, сельском хозяйстве, при чрезвычайных ситуациях и т. д. Наиболее востребованными являются карты местности, построенные на основе данных дистанционного зондирования Земли (ДЗЗ). В зависимости от решаемых задач используются снимки низкого, среднего или высокого разрешения. Однако для получения нужной информации в дополнение к широким возможностям выбора и получения ДЗЗ необходимо наличие быстрых, надежных методов и алгоритмов дешифрирования.

При решении упомянутых и многих других прикладных задач используются снимки среднего или низкого пространственного разрешения с большим количеством спектральных каналов, которые были получены, например, со спутников Landsat 5, 7, 8 и др. [1]. Задачи сегментации и дешифрирования таких спектральных снимков трудны: для их решения требуется большой объем работы, выполняемой специалистами вручную [2]. Трудность здесь вызвана невысоким пространственным разрешением снимков; например, для спутников Landsat 7 и 8 оно равно 30 м на пиксел для спектральных каналов и 15 м на пиксел для панхроматического канала. Разработка алгоритмов выделения объектов различного типа на таких изображениях до сих пор является актуальной задачей.

К сожалению, алгоритмы автоматической сегментации не отличаются высокой точностью и надежностью, поэтому интерактивные методы с различной долей автоматизации [3–12] по-прежнему остаются более предпочтительными для выполнения практического дешифрирования аэрофотоснимков и космических изображений среднего и низкого разрешения.

В процессе векторизации космических снимков значительные усилия затрачиваются на выделение линейных объектов: дорог, рек, а также границ объектов. При этом независимо от того, является линейный объект асфальтной или проселочной дорогой, или тропинкой, или малой рекой, для снимков среднего и низкого пространственного разрешения его ширина обычно равна одному или нескольким пикселам. Это часто делает невозможным выполнение задачи векторизации снимка в автоматическом режиме и существенно затрудняет ее решение в интерактивном.

В настоящей работе предлагается алгоритм интерактивного выделения линейных объектов на аэрофотоснимках и космических изображениях среднего и низкого пространственного разрешения с помощью ломаных линий. Преимуществами предложенного алгоритма является возможность его быстрого выполнения в течение долей секунды, легкая настройка для практической работы, возможность удобной ручной корректировки решения. Интерактивная часть алгоритма заключается в задании экспертом на изображении начальной и конечной точек вы-

деляемого участка линейного объекта. Приводится оценка быстродействия последовательной реализации алгоритма, а также описывается алгоритм, основанный на разработанной версии метода динамического программирования [13], позволяющий проводить вычисления в параллельном режиме, тем самым уменьшая количество операций при поиске наилучшего решения и объем используемой при этом памяти. Описываются результаты, полученные при практической параллельной реализации данного алгоритма с использованием технологии программирования видеокарт CUDA [14, 15].

1. Описание алгоритма выделения линейных объектов

Предложенный алгоритм приближает ломаными линиями наперед заданной кривизны участки линейных объектов на изображениях, которые ограничены двумя пикселами, выделенными экспертом в интерактивном режиме (рис. 1).

Входными параметрами алгоритма являются:

– начальный a и конечный b пикселы участка линейного объекта;

– число звеньев ломаной;

– длина направляющих отрезков, на которых выбираются узлы ломаных (см. рис. 1);

– допустимый угол γ между соседними звеньями ломаной, ограничивающий кривизну решения.

Наряду с указанными входными параметрами можно также изменять в случае необходимости ширину в пикселах растрового представления ломаной, а также выбирать тип градиентного оператора, предназначенного для выделения границ объектов на изображении, или даже задавать его в явном виде.

Выполнение алгоритма начинается с ручного выделения двух пикселов a и b, ограничивающих участок линейного объекта, который затем будет найден автоматически. Выделенные пикселы соединяются построенными алгоритмом непрерывными ломаными линиями, узлы которых $a = p_0, p_1, ..., p_n = b$ лежат на направляющих, перпендикулярных отрезку $[a, b]$ (см. рис. 1). Среди всех ломаных выбирается та, которая в наибольшей мере заполнена градиентом изображения, ортогональным к звеньям этой ломаной. Подробное описание нескольких функций-критериев заполненности ломаной градиентом приведено в [16]. Найденная ломаная принимается в качестве приближения участка линейного объекта.

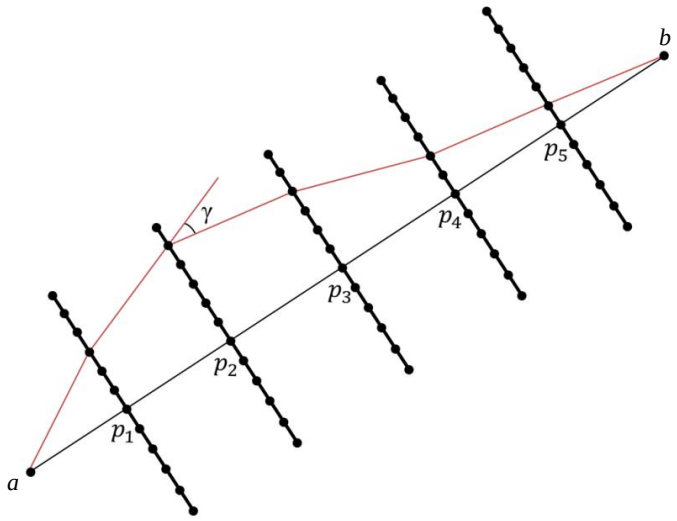

Рис. 1. Пример расчета частичных сумм с учетом угла между отрезками γ

Формально предложенный алгоритм состоит из следующих шагов, которые поясняет рис. 1:

Шаг 1. Пользователь задает исходные данные: начальный *a* и конечный *b* пикселы участка линейного объекта на аэрофотоснимке или космическом изображении, который будет при-

ближен ломаной линией; число звеньев ломаной n; чисто точек на каждой направляющей d; максимально разрешенный угол излома ломаной γ, ограничивающий допустимую кривизну решения.

Шаг 2. Строится отрезок прямой с концами a и b, который разбивается на заранее заданное пользователем количество одинаковых отрезков. Координаты пикселов разбиения заносятся в массив.

Шаг 3. Выбирается один из известных операторов для построения градиента исходного изображения, например оператор Превитт, Собеля и т. д., или желаемый оператор задается пользователем интерактивно в зависимости от характеристик обрабатываемого изображения.

Шаг 4. Вычисляется градиент исходного изображения, который сохраняется в отдельном массиве.

Шаг 5. Для пикселов $a = p_0, p_1, ..., p_n = b$ (сохраненных в массиве), разбивающих отрезок $[a, b]$ на равные интервалы, строятся направляющие ломаных – отрезки прямых линий заданной длины, перпендикулярные $[a, b]$ и проходящие через p_j, $j = 1, ..., n-1$. Каждая направляющая разбивается на равные интервалы, число которых задается пользователем. Имеет смысл разбивать направляющие так, чтобы длина интервалов была больше одного пиксела.

Шаг 6. Рассматриваются всевозможные растровые ломаные с началом в пикселе a, концом в пикселе b и узлами в концах интервалов разбиения направляющих. Для каждой растровой ломаной вычисляется и сохраняется сумма S модулей синусов углов между градиентом изображения и отрезком ломаной.

Шаг 7. Находится максимальная сумма $S_{\max}$ из всех сумм, вычисленных на шаге 6 при полном переборе всех ломаных с учетом задаваемого пользователем максимального допустимого угла γ между отрезками двух соседних направляющих. Запоминаются также координаты узлов этой ломаной.

Шаг 8. Записываются в массив координаты узлов найденной ломаной с $S = S_{\max}$.

Шаг 9. Найденная ломаная линия, которая принимается за приближение участка линейного объекта, отображается на фоне исходного изображения.

Шаг 10. В случае необходимости пользователь может скорректировать форму ломаной в интерактивном режиме путем перетаскивания одного или нескольких ее узлов.

Шаг 11. Полученное решение принимается или отменяется пользователем. Если результат принят, его можно сохранить в файл и продолжить работу алгоритма. В случае продолжения пиксел b становится начальным для нового участка линейного объекта, т. е. полагается $a = b$. Далее выбирается новый пиксел b (конец нового участка) и повторяются шаги 1–10.

2. Оценка быстродействия последовательной реализации алгоритма

При дешифрировании изображений программная реализация алгоритма запускается многократно, поэтому одним из основных требований к ней является возможность выполнения алгоритма в режиме реального времени либо не более чем с односекундной задержкой.

Тестирование неоптимизированной версии алгоритма, основанной на полном переборе всех ломаных, показало, что большая часть времени вычисления приходится на выбор наилучшей ломаной из всех построенных (рис. 2). Оно очень велико для практической реализации алгоритма.

Количество операций N при полном переборе всех вариантов зависит от числа всех возможных ломаных. Очевидно, что если d – число точек на каждой направляющей, n – их количество, то $N = O\left(d^n\right)$. Например, время вычисления оптимальной ломаной с 7 узлами и 64 точками на каждой направляющей методом полного перебора на CPU Core2Quad Q6600 равно 77,6 с.

Не слишком большой эффект дает распараллеливание алгоритма в чистом виде для выполнения на двух-четырех ядрах процессора. При этом не остается свободных ресурсов процессора для выполнения других задач.

Для ускорения расчетов необходимо было, во-первых, переработать алгоритм, уменьшив количество операций, необходимых для поиска оптимального решения, и, во-вторых, макси-

мально распараллелить его программную реализацию, задействовав видеокарту, которая обеспечивает одновременное выполнение огромного числа операций.

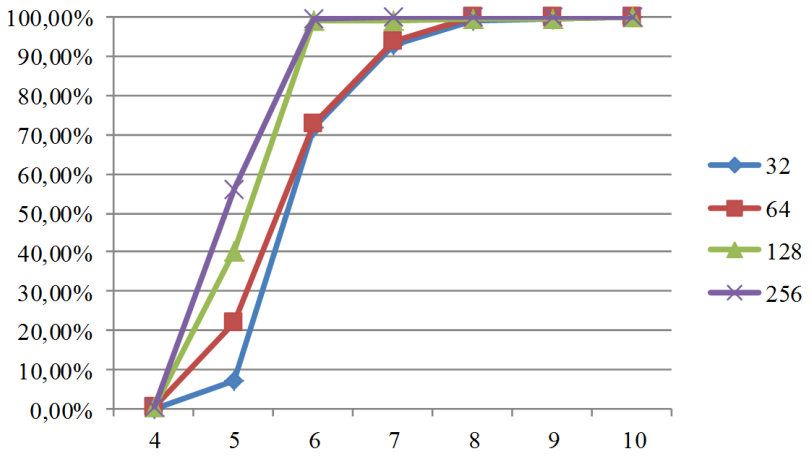

Рис. 2. Доля времени поиска оптимальной ломаной среди всех допустимых, в процентах от общего времени работы алгоритма. По оси *X* отложено число узлов ломаной. Кривые соответствуют 32, 64, 128, 256 узлам на каждой направляющей

Если не учитывать ограничение на кривизну ломаной, методы динамического программирования (ДП) позволяют найти наилучшее приближение за $N = O\left(d^2(n-1)+d\right)$ операций, где, как и ранее, d – число точек на каждой направляющей, n – количество направляющих. Однако для выделения линейных объектов с помощью ломаных предпочтительно использовать ограничение на их кривизну, так как это позволяет значимо повысить точность выделения на изображениях невысокого разрешения и при наличии шумов и искажений. Специально для такого случая была разработана версия метода динамического программирования, позволяющая находить приближения ломаными заданной кривизны за $N = O\left(d^3(n-2)+d^2\right)$ операций.

Для дальнейшего ускорения вычислений были разработаны параллельные реализации ДП-версий алгоритма с использованием технологии программирования CUDA.

3. Общее описание параллельного алгоритма поиска наилучшего решения

Параллельная версия алгоритма специально разработана для большого количества одновременно работающих потоков с учетом трех основных требований: сократить время выполнения и общее количество операций, а также уменьшить объем памяти, используемой на промежуточных шагах работы алгоритма. Ее особенность заключается в возможности параллельного выполнения шагов алгоритма. Эффективное распараллеливание алгоритма возможно в случае, когда функция-критерий алгоритма, на основе которой строится ломаная, приближающая линейный объект, является аддитивной функцией относительно звеньев ломаной.

Обозначим через $\varphi(L)$ функцию-критерий – числовую характеристику ломаной L, на основе максимизации которой строится приближение (детальное описание критериев приведено в [16]). Будем использовать $\varphi(L)$, аддитивные относительно ее звеньев:

$$\varphi(L) = \sum_{i=1}^{n} \varphi\left(L(j_{i-1}, j_i)\right),$$

где $j_0 = a$, $j_n = b$ – начальная и конечная точки ломаной;

$j_i = 1, 2, ..., d\ (i = 1, 2, ..., n-1)$ – координаты узлов ломаной относительно соответствующей направляющей (см. рис. 1), или, иными словами, номера точек на каждой направляющей;

$L(j_{i-1}, j_i)$ – отрезки ломаных, ограниченные точками j_{i-1}, j_i.

Для простоты поясним принцип работы алгоритма на примере ломаной с пятью направляющими линиями (рис. 3). Координаты внутренних звеньев ломаных относительно соответствующих направляющих будем обозначать i, j, k, l, m.

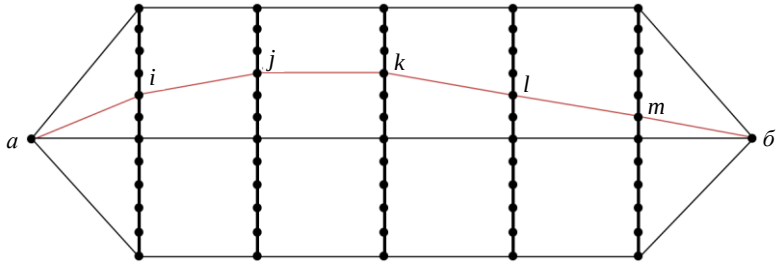

Рис. 3. Пример построения направляющих линий в точках изгиба

Шаг А. Вычисляем и запоминаем характеристики $\varphi\big(L(j_{i-1}, j_i)\big)$ всех возможных отрезков ломаных.

Шаг Б. Вычисляем характеристики $\varphi = \varphi\big(L(i, j, k)\big)$ всех трехзвенных ломаных допустимой кривизны $L(i, j, k)$, выходящих из точки a и проходящих через точки i, j в каждую точку k третьей направляющей линии. Для каждой пары индексов j, k второй и третьей ломаных находим и сохраняем значения функционала $\varphi\big(L(i, j, k)\big)$ на локально оптимальных трехзвенных ломаных, проходящих через точки j, k, и номер точки на первой направляющей для оптимальной ломаной:

$$\varphi_{j,k}(1) = \max_i{}^{*}\left\{\varphi\big(L(i, j, k)\big)\right\} \text{ и } \tilde{i}_{j,k}(1) = \arg\max_i{}^{*}\left\{\varphi\big(L(i, j, k)\big)\right\}. \quad (1)$$

Символ $*$ в формуле (1) обозначает, что максимум берется только по трехзвенным ломаным с допустимой кривизной, иначе говоря, таким, у которых соседние отрезки образуют между собой угол, меньший наперед заданного угла γ.

Шаг В. Вычисляем и запоминаем значения функционала $\varphi\big(L(i, j, k, l)\big)$ на локально оптимальных четырехзвенных ломаных, проходящих через точки k, l:

$$\varphi_{k,l}(2) = \max_{i,j}{}^{*}\left\{\varphi\big(L(i, j, k, l)\big)\right\} = \max_j{}^{*}\left\{\varphi_{j,k}(1) + \varphi\big(L(k, l)\big)\right\}$$

и значения координат

$$\tilde{i}_{k,l}(2), \tilde{j}_{k,l}(2) = \arg\max_{i,j}{}^{*}\left\{\varphi\big(L(i, j, k, l)\big)\right\}.$$

Шаг Г. Повторяем вычисления для пятизвенных ломаных $L(i, j, k, l, m)$:

$$\varphi_{l,m}(3) = \max_{i,j,k}{}^{*}\left\{\varphi\big(L(i, j, k, l, m)\big)\right\} = \max_k{}^{*}\left\{\varphi_{k,l}(2) + \varphi\big(L(l, m)\big)\right\}$$

и

$$\tilde{i}_{l,m}(3), \tilde{j}_{l,m}(3), \tilde{k}_{l,m}(3) = \arg\max_{i,j,k}{}^{*}\left\{\varphi\big(L(i, j, k, l, m)\big)\right\}.$$

Шаг Д. Аналогично предыдущим вычислениям находим

$$\varphi_m(4) = \max_{i,j,k,l}{}^{*}\left\{\varphi\big(L(i, j, k, l, m, b)\big)\right\} = \max_l{}^{*}\left\{\varphi_{l,m}(3) + \varphi\big(L(m, b)\big)\right\}$$

и

$$\tilde{i}_m(4),\ \tilde{j}_m(4),\ \tilde{k}_m(4),\ \tilde{l}_m(4) = \arg\max^{*}_{i,j,k,l} \left\{ \varphi\left(L(i,\ j,\ k,\ l,\ m,\ B) \right) \right\}.$$

Шаг Е. Вычисляем максимум по m среди чисел $\varphi_m(4)$ и соответствующие значения узлов, находим координаты $\tilde{i}(5),\ \tilde{j}(5),\ \tilde{k}(5),\ \tilde{l}(5),\ \tilde{m}(5)$ звеньев оптимальной ломаной.

Шаг Ж. Конец.

Данная версия алгоритма допускает быстрое параллельное выполнение на современных бюджетных видеокартах, таких, например, как GeForce GTX 650 Ti.

Параллельная версия алгоритма, по сути, заключается в выполнении шагов 1–5 последовательного алгоритма, шагов A–E параллельной версии вычисления максимума функции-критерия и затем шагов 8–11 последовательной версии.

4. Результаты тестирования алгоритма и его параллельной реализации

Тестирование показало достаточно высокую точность построенного алгоритма (рис. 4).

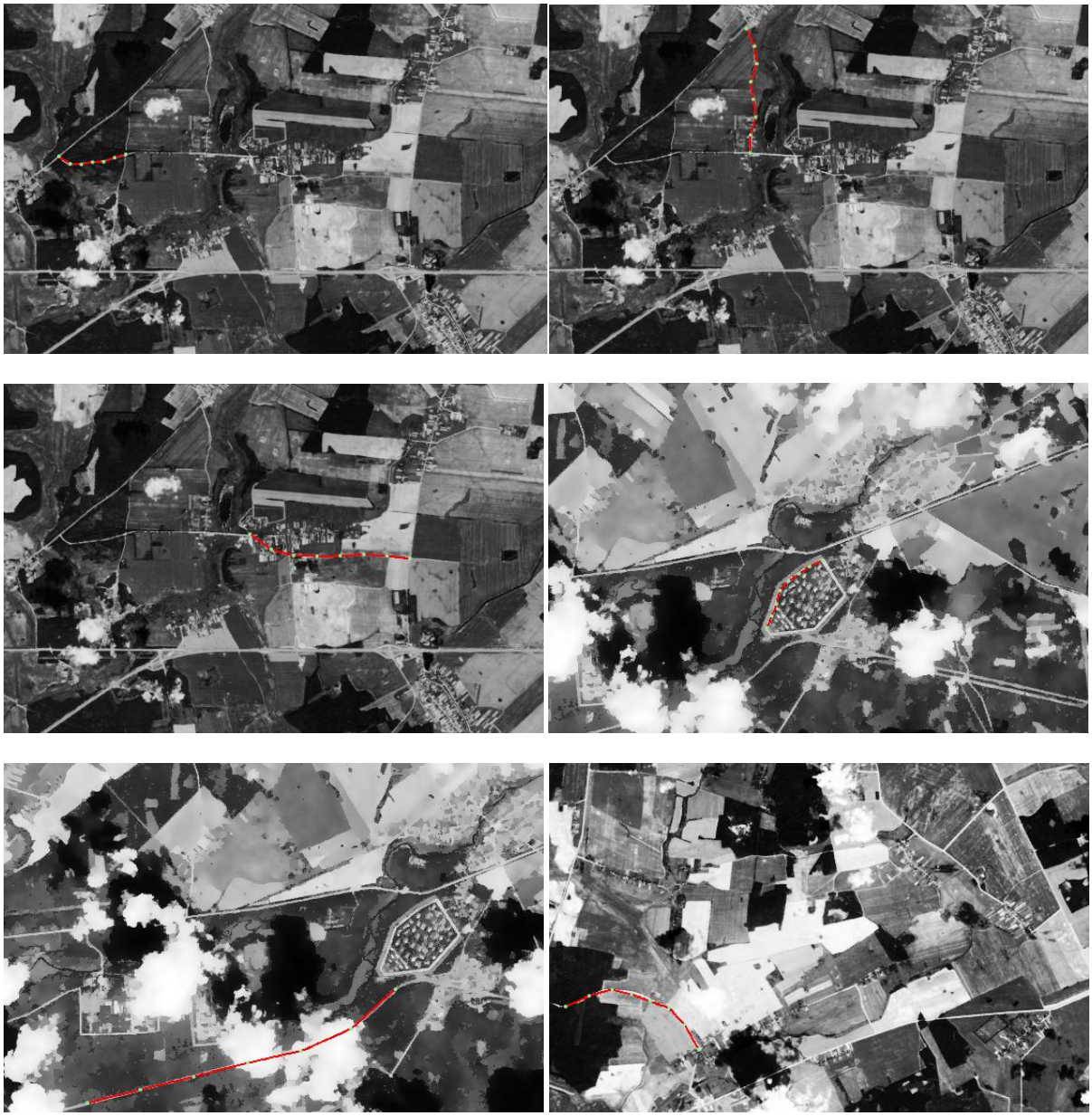

Рис. 4. Примеры работы алгоритма

На космических снимках, полученных со спутника Landsat 8, алгоритм выделяет более 95 % участков линейных объектов. После его настройки на конкретные снимки выделяются остальные 5 %.

Параллельная версия алгоритма реализована с использованием технологии программирования CUDA. Полученные данные тестирования (таблица) показывают, что CUDA-версия параллельного алгоритма выполняет расчеты за приемлемое время (доли секунды) в широком диапазоне значений входных параметров. При этом время расчетов не сильно изменяется при увеличении количества операций.

Время выполнения на GPU GeForce GTX 650Ti параллельной версии функции поиска наилучшего решения без ограничения угла между соседними ломаными линиями для 32, 64, 128, 256 точек на направляющих, мс

Количество интервалов разбиения m	Количество точек на направляющих линиях d			
	32	64	128	256
5	203	218	234	249
6	218	234	249	265
7	234	234	249	265
8	234	234	265	297

Заключение

В работе предложен алгоритм интерактивного выделения линейных объектов на аэрофотоснимках и космических изображениях среднего и низкого пространственного разрешения. Его интерактивная часть заключается в выделении экспертом начального и конечного пикселов участка линейного объекта, намеченного для выделения. Далее алгоритм автоматически строит непрерывную ломаную линию наперед заданной кривизны с концами в определенных экспертом пикселах, которая приближает выделенный участок. После окончания работы алгоритма эксперт может скорректировать построенную ломаную, перемещая ее узлы мышкой.

Преимуществами предложенного алгоритма является возможность его быстрого выполнения в течение долей секунды, легкая настройка для практической работы, возможность удобной ручной корректировки решения.

В статье приведены оценки быстродействия последовательной реализации алгоритма. Реализация алгоритма интерактивного выделения линейных объектов на аэрофотоснимках и космических изображениях среднего и низкого пространственного разрешения основана на разработанной версии метода динамического программирования и позволяет проводить вычисления в параллельном режиме. Разработанные подходы к реализациям предложенного алгоритма позволяют уменьшить на несколько порядков объем требуемых вычислений и сократить время его выполнения. Также даны оценки быстродействия разработанной параллельной версии алгоритма. Описаны результаты, полученные при практической реализации данного алгоритма для процессора и с использованием технологии программирования видеокарт CUDA.

Список литературы

1. LandsatLook Viewer [Электронный ресурс]. – Mode of access : http://landsatlook.usgs.gov. – Date of access : 28.09.2014.

2. Кочуб, Е.В. Анализ методов обработки материалов дистанционного зондирования Земли / Е.В. Кочуб, А.А. Топаз // Вестник ПГУ. Сер. F. – 2012. – № 16. – С. 132–140.

3. Supreet, S. Automatic Road Detection of Satellite Images – A Survey / S. Supreet, B. Seema // Intern. J. of Computer Applications & Information Technology. – 2013. – Vol. 3(2). – P. 32–34.

4. Kalaivanan, R. Survey on Road Extraction From High Resolution Satellite Images / R. Kalaivanan, S. Mishmala // Intern. J. of Advanced Research in Computer and Communication Engineering. – 2013. – Vol. 2(10). – P. 4156– 4159.

5. A Family of Quadratic Snakes for Road Extraction / M.N. Dailey [et al.] // Lecture Notes in Computer Science. – 2007. – Vol. 4843. – P. 85–94.

6. Dal Poz, A.P. Dynamic Programming Approach For Semi-Automated Road Extraction From Medium- And High-Resolution Images / A.P. Dal Poz, G.M. do Vale // ISPRS Archives. – Vol. 34 (3/W8). – P. 87–91.

7. Urban digital map updating from satellite high resolution images using GIS data as a priori knowledge / T. Bailloeul [et al.] // Remote Sensing and Data Fusion over Urban Areas, 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas. – Urban, 2003. – P. 283–287.

8. Niu, X. A semi-automatic framework for highway extraction and vehicle detection based on a geometric deformable model / X. Niu // Photogrammetric Engineering and Remote Sensing. – 2006. – Vol. 61. – P. 170–186.

9. Automatic Road Extraction from Satellite Imagery Using LEGION Networks / J. Yuan [et al.] // Proc. of Intern. Joint Conference on Neural Networks. – Atlanta, Georgia, USA, 2009. – P. 3471–3476.

10. Lacoste, C. Unsupervised line network extraction in remote sensing using a polyline process / C. Lacoste, X. Descombes, J. Zerubia // Pattern Recognition. – 2010. – Vol. 43 (4). – P. 1631–1641.

11. Color image segmentation: advances and prospects / H. Cheng [et al.] // Pattern Recognition. – 2001. – № 34. – P. 2259–2281.

12. Sniedovich, M. Dynamic programming. Foundations and principles / M. Sniedovich. – Boca Raton : CRC Press Taylor & Francis Group, 2011.

13. Handbook of Learning and Approximate Dynamic Programming / J. Si [et al.] // Wiley-IEEE Press, 2004.

14. Боресков, А.В. Основы работы с технологией CUDA / А.В. Боресков, А.А. Харламов. – М. : ДМК Пресс, 2010. – 232 с.

15. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессоров / Дж. Сандерс, Э. Кэндрот. – М. : ДМК Пресс, 2011. – 232 с.

16. Zalesky, B.A. Interactive extraction of roads and rivers in low resolution or noisy satellite images / B.A. Zalesky, E.N. Seredin // Proc. of 12th Intern. Conf. PRIP2014. – Minsk, 2014. – P. 329–334.

Поступила 02.06.2014

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: eduard.seredin@tut.by

E.N. Seredin, B.A. Zalesky

INTERACTIVE ALGORITHM FOR SELECTION OF LINEAR OBJECTS ON AERIAL PHOTOGRAPHS AND SATELLITE IMAGES

A new approach to the problem of extraction of linear objects on aerial photographs and satellite images is presented. An interactive algorithm of detection of linear objects by broken lines, based on the developed approach, is offered. The algorithm provides fast detection of linear objects,it is easy tuned and convenient for practical use. It is enough precise and reliable to process middle or low resolution aerial photographs and satellite images. Specificity of its parallel program implementation is also described. The developed parallel program implementation allowed to reduce the execution time of the algorithm by several orders. Estimates of execution time of versions of the algorithm, which are intended for video cards, are given in the article.