Современные проблемы дистанционного зондирования Земли из космоса. 2013. Т. 10. № 4. С. 98–106

Метод вычислительной оптимизации в задаче сопоставления растровой и векторной информации при анализе спутниковых данных

В.Г. Бондур [1], А.Б. Мурынин [1, 2], И.А. Матвеев [1, 2], А.Н. Трёкин [1, 3], И.А. Юдин [1]

[1] Научно-исследовательский институт аэрокосмического мониторинга «АЭРОКОСМОС»
Москва, Россия
E-mail: amur@ccas.ru
[2] Вычислительный центр им. А.А. Дородницына Российской академии наук
Москва, Россия
E-mail: matveev@ccas.ru
[3] Московский физико-технический институт (государственный университет)
Долгопрудный, Моск. обл., Россия
E-mail: alexey.trekin@gmail.com

В различных задачах дистанционного зондирования Земли применяется обработка космических снимков, представленных как в растровом, так и в векторном форматах, поэтому возникает задача сопоставления разнородных данных. Предлагается простой метод пространственной индексации векторных данных, который увеличивает скорость поиска объектов в векторном файле и таким образом ускоряет совместную обработку данных. Предложенный метод индексации сравнивается с существующими, проводятся теоретический расчет сложности алгоритма и вычислительные эксперименты, подтверждающие его эффективность.

Ключевые слова: индексация, обработка изображений, дистанционное зондирование Земли.

Введение

Дистанционное зондирование Земли (ДЗЗ) – эффективный инструмент, с помощью которого можно решать различные научные и практические задачи. Большая часть информации, полученной методами ДЗЗ, хранится в двух различных типах данных – растровом и векторном. Растровая информация отображает поверхность Земли как массив пикселей, каждый из которых соответствует некому участку поверхности. Именно в растровом формате поставляется исходная информация со спутников космической фотосъемки. Векторные файлы обычно содержат информацию о расположении или геометрических характеристиках какого-либо класса объектов в виде набора записей в терминах двумерной системы координат «широта–долгота». Как правило, векторные файлы формируются на основе растровых данных и представляют собой совокупность полигонов – геометрических протяженных объектов на поверхности Земли. Зачастую возникает необходимость работать одновременно с растровыми и векторными данными об одной и той же территории и сопоставлять их. Например, при обработке данных о природных пожарах (Бондур и др., 2012) важной информацией является карта выгоревших территорий, получаемая по совокупности большого количества распределенных во времени наблюдений, при которых установлены места активного горения. Карта выгоревших территорий получается в векторном виде, в то время как каждое изображение территории, используемое при ее построении, равно как и новые обрабатываемые изображения, представлены растрами.

Постановка задачи

Для каждого полигона $P = \left\{ \left(x_j ; y_j \right)_{j=1}^{J} \right\}$ в векторном файле требуется определить, какие пиксели (X, Y) растрового файла лежат внутри него. Координаты векторного файла будем считать географическими. Растровый файл является геопривязанным, т.е. для каждого пикселя известны его географические координаты, иными словами задано отображение $G : (X, Y) \xrightarrow{G} (x, y)$. Эту задачу можно решать двумя путями: прямым и обратным. Обратный путь состоит в том, чтобы для каждого полигона определить пиксели растрового изображения, лежащие внутри него. Для этого требуется построить отображение $H = G^{-1} : (x, y) \xrightarrow{H} (X, Y)$. В этом случае было бы возможно применение алгоритмов, ориентированных на выбор простой процедуры, описывающей окрестности векторного объекта на растровом изображении, и перебор ограниченного числа пикселов для проверки их пересечения с векторным объектом. Однако отношение *G* является немонотонным и разрывным в областях проекционных искажений (так называемый bow-tie effect (Wen, 2008)), что приводит к невозможности построения отображения *H* и неприемлемому возрастанию перебора при поиске.

Поэтому для решения проблемы был выбран прямой путь: для каждого обрабатываемого пикселя растра (X, Y) взять географические координаты и найти в векторном файле все полигоны P_i, включающие точку с этими координатами $P_i : (x, y) = G(X, Y) \in P_i$. Проверка нахождения точки в границах одного полигона может осуществляться, например, методом трассировки луча с подсчетом количества пересечений (Haines, 1994). Для проверки всех полигонов требовалось бы провести трассировку луча по всему их набору, что неприемлемо для реальных задач, включающих десятки тысяч полигонов. Таким образом, необходимо ввести некоторую структуру, упорядочивающую полигоны и уменьшающую перебор. В этом случае работа алгоритма сопоставления состоит из двух шагов: построение индекса (*индексация*) и собственно сопоставление с использованием индекса.

Для подобных задач разработано множество методов индексации, таких как различные варианты R-деревьев (R*, R+, Priority R, X- деревья). Эти достаточно сложные структуры позволяют быстро решать разнообразные трудоемкие задачи взаимного расположения пространственных объектов. Основой для всех этих методов является заключение объектов пространства в ограничивающий прямоугольник и построение сбалансированного, сильно разветвленного дерева, в котором каждая вершина представляет прямоугольник, содержащий в себе все объекты, размещенные в ее потомках, а листовая вершина представляет собой ограничивающий прямоугольник одного объекта. Первый из этих методов, R-дерево, способен ускорять решение множества задач на взаимное расположение многомерных протяженных объектов (Guttman, 1984). Близок к исследуемой задаче метод дерева квадрантов (Finkel, Bentley, 1974). В нем заранее выбирается допустимая емкость листовой вершины – наибольшее количество объектов в ней, а внутренняя вершина дерева всегда имеет четыре потомка, представляющих квадранты (четверти) области родительской вершины. С учетом этого при добавлении в дерево новой записи она либо добавляется в листовую

вершину, соответствующую ее координатам, если в этой вершине есть еще неиспользованная емкость, либо, если вершина заполнена, она становится внутренней, а все объекты, отнесенные к ней, распределяются между четырьмя ее потомками. Однако при решении простейшей задачи – взаимного расположения точки и набора непересекающихся полигонов – возможно применение существенно более простого метода индексации, который в этом случае дает лучший результат, чем универсальные алгоритмы.

Описание метода

Пусть задан векторный файл, содержащий полигоны P_i, заданные как последовательности вершин в координатах широта–долгота $P_i = \left\{\left(x_{i,j}; y_{i,j}\right)_j\right\}_i$. Полигоны построены на основании некоторого ранее обработанного растрового изображения (или нескольких) и описывают один пиксель изображения или объединение нескольких. Поэтому большинство полигонов имеют размер одного порядка и не являются сильно вытянутыми, т.е. отношение длин любых двух сечений полигона порядка единицы. Характерный размер (средний диаметр) полигона обозначим $l = \langle diam\, P_i \rangle$. Общий размер области, содержащей полигоны, обозначим $\left(L_x; L_y\right)$.

Построение индексной структуры производится следующим образом. Для всех записей векторного файла ищется общий ограничивающий прямоугольник L, размера $\left(L_x; L_y\right)$: $L_x = \max\limits_{i,j} x_{i,j} - \min\limits_{i,j} x_{i,j}$, $L_y = \max\limits_{i,j} y_{i,j} - \min\limits_{i,j} y_{i,j}$, который делится на некоторое количество равных непересекающихся прямоугольников-ячеек $C_{u,v}$ таким образом, чтобы каждая ячейка имела наиболее близкую к квадрату форму. Размеры ячеек (соответственно, их количество) выбираются из соображений минимизации общего времени вычислений, более подробно описанных ниже. Затем для каждой ячейки создается список полигонов, пересекающих ячейку (иными словами полностью или частично содержащихся в ней): $Q_{u,v} = \left\{P_i \mid P_i \cap C_{u,v} \neq \varnothing\right\}$. Координаты ячеек и списки номеров полигонов записываются, на этом построение индексной структуры заканчивается. На *рис. 1* показан пример разбиения на ячейки и порождаемые списки контуров для ячеек. Полигон A пересекается с ячейками $C_{1,1}$, $C_{1,2}$, $C_{2,1}$; полигон B – с ячейками $C_{1,1}$ и $C_{2,1}$; полигон D полностью лежит в ячейке $C_{1,1}$. Соответственно, списки полигонов для ячеек: $Q_{1,1} = \{A, B, D\}$, $Q_{1,2} = \{A\}$, $Q_{2,1} = \{A, B\}$, $Q_{2,2} = \varnothing$.

На этапе сопоставления, когда нужно выяснить, лежит ли точка с определенными координатами в каком-либо из полигонов векторного файла, по этим координатам явным образом вычисляется номер ячейки, в которой лежит точка, и затем происходит перебор лишь полигонов, находящихся в списке для данной ячейки, с проверкой, лежит ли точка в каком-либо из них.

Следует отметить важное упрощение, сделанное на шаге индексации. Известно, что задача определения того, пересекается ли полигон с прямоугольником, решается за $O(n)$ шагов, где n – число вершин полигона (Agoston, 2004). В случае большого количества вершин полигона и большого его размера это может привести к квадратичному $O(n^2)$ возрастанию сложности вычислений за счет возрастания числа ячеек, которые могут пересекаться

с полигоном. На практике такие большие полигоны появляются достаточно часто. Однако для решаемой задачи не является критичным добавление в список ячейки полигона, даже если он не пересекается с ней. В этом случае на этапе сопоставления будут затрачены дополнительные излишние вычисления при обработке точек ячейки, не пересекающейся с данным полигоном, но ошибочно содержащей его в своем списке. Порядок сложности избыточных вычислений равен $O(n)B_{u,v}$, где $B_{u,v}$ – число точек в ячейке, что может оказаться существенно меньше затрат $O(n^2)$ на этапе индексации.

Рис. 1. Пример векторного файла, разбитого на ячейки

Поэтому для установления факта пересечения полигона с ячейкой используется не обсчет самого полигона P_i, а сравнение ячейки с ограничивающим прямоугольником этого полигона: $(L_x; L_y)_i$, $L_{x,i} = \max_j x_{i,j} - \min_j x_{i,j}$, $L_{y,i} = \max_j y_{i,j} - \min_j y_{i,j}$. Таким образом, вместо сложного алгоритма определения пересечения прямоугольника с произвольным полигоном используется простейшее определение пересечения двух прямоугольников со сторонами, параллельными осям координат. На *рис. 2* приведен пример ошибочного включения полигона A (показанного на *рис. 1*) в список $Q_{2,2}$.

Оценим вычислительную сложность метода. Пусть L_x – горизонтальный размер обрабатываемых данных по оси OX (географические координаты), т.е., фактически, разброс по долготе; L_y – вертикальный размер обрабатываемых данных по оси OY (географические координаты), т.е., фактически, разброс по широте; K – число полигонов; n – среднее число вершин в полигоне; $N = nK$ – общее число вершин во всех полигонах; l – средний линейный размер полигона (географические координаты); a – размер ячейки (географические координаты); B – число обрабатываемых точек.

Рис. 2. Пример ошибочного включения ячейки в список пересекающихся с полигоном

При расчете без индексации алгоритмическая сложность равна произведению числа обрабатываемых точек растра на число полигонов и на среднее число вершин в полигоне: $A = B\beta N$, где $\beta < 10$ – заранее заданное ограниченное число. При индексации необходимо оценить сложность самой индексации и сложность обработки точки в зависимости от размера ячейки.

Шаг индексации

Оценка среднего количества ячеек, которые пересекает полигон, равно отношению их площадей: $O\left(l^2 / a^2\right)$. Общая оценка сверху – $\max\left\{1; \alpha\pi\, l^2 / a^2\right\}$, где $\alpha < 2$ – коэффициент для оценки площади полигона. Отметим, что оценка $O\left(l^2 / a^2\right)$ справедлива только при условии $l >> a$, а при условии $l << a$ среднее количество ячеек близко к 1. При $l \approx a$ рассмотрение затруднительно. Алгоритмическая сложность всего шага индексации: $A_1 = \max\left\{K; \alpha\pi \dfrac{l^2}{a^2} K\right\}$.

Шаг сопоставления по индексу

Среднее число полигонов, пересекающих ячейку, установим из того, что число пересечений всех полигонов со всеми ячейками равно числу пересечений всех ячеек с полигонами, и равно A_1. Обозначим q – искомое среднее число полигонов, пересекающих ячейку. Всего ячеек $L_x L_y / a^2$, поэтому $qL_x L_y / a^2 = A_1$, откуда получаем:

$$q = A_1 \frac{a^2}{L_x L_y} = \begin{cases} \dfrac{a^2}{L_x L_y} K & \text{при } l >> a \\ \alpha\pi \dfrac{l^2}{L_x L_y} K & \text{при } l << a \end{cases}.$$

Так как обрабатывается список длиной B точек и проверка, лежит ли точка внутри полигона, имеет сложность $O(n)$, получаем сложность сопоставления:

$$A_2 = qO(n)B = O(N)B\begin{cases} \dfrac{a^2}{L_x L_y} & \text{при } l >> a \\ \alpha\pi \dfrac{l^2}{L_x L_y} & \text{при } l << a. \end{cases}$$

Сложность метода в целом: $A = A_1 + A_2 = A_1 + O(n)BA_1 \dfrac{a^2}{L_x L_y} = A_1\left(O(n)B\dfrac{a^2}{L_x L_y} + 1\right)$.

Оценим также объем памяти, необходимый для содержания структуры. При размере ячейки $a = l$ количество ячеек, а, значит, объем памяти, необходимый структуре, оценивается как $O([L_x / l], [L_y / l])$, т.е. пропорционален площади области векторного файла и обратно пропорционален характерной площади полигона.

На практике размеры полигонов различаются, поэтому действительно оптимальный размер ячейки может дать только экспериментальная проверка, но так или иначе, он должен быть одного порядка с характерным размером полигона.

Экспериментальная проверка

Проверка работы данного алгоритма и определение его быстродействия были проведены с использованием действительных данных дистанционного зондирования. В качестве векторных файлов использовались векторные файлы (шейпфайлы), содержащие полигоны – территории, пострадавшие от природных пожаров, сформированные автоматически на основе космических изображений MODIS с разрешением 1 км. Точки земной поверхности, положение которых по отношению к полигонам векторного файла проверялось, брались из файла другого изображения MODIS, содержащего растровый снимок территории, пересекающей область векторного файла.

Средняя площадь зоны активного горения лесного пожара составляет менее квадратного километра (Бондур, 2010), на изображении детектированный пожар отображается чаще всего одним пикселем, и при формировании векторных данных представляется в виде прямоугольника. Исходя из оценки алгоритмической сложности, наименьшее время работы должно быть при размерах ячейки, близких размеру пикселя. Во всех исследованных векторных файлах, содержащих реальные спутниковые данные об активных пожарах, средний размер полигона составил 0,015–0,042 градуса. При этом средние размеры по широте и по долготе различаются не более чем в 3 раза, значит ограничение по слабой вытянутости полигонов также справедливо.

В исследовании первоначальный размер ячейки выбран равным одному градусу географических координат, и на каждой следующей итерации уменьшался вдвое, пока время работы не достигало локального минимума (т.е. точки, когда время в следующей итерации оказывалось уже больше), или до тех пор, когда создание индексной структуры становилось невозможным из-за ограничений памяти.

На *рис. 3* приведены графики зависимости времени работы алгоритма *T* и используемой оперативной памяти *V* от размера ячейки *D* на векторном файле, содержащем обнаруженные по космическим изображениям природные пожары за различные периоды 2010 г. (18 486 полигонов, файл test2.shp). Шкала по обеим осям логарифмическая. Видно, что при уменьшении размера ячейки от большого (1 градус, т.е. порядка 50 линейных размеров полигона) до размеров менее 0,1 градуса время падает, но затем при уменьшении размеров до значений, меньших характерного размера полигона, растет. Одновременно с этим при уменьшении размера ячейки монотонно растет объем занимаемой памяти. Учитывая, что характерный диаметр полигона *l* в данных файлах составляет 0,01–0,02 градуса, оптимальное время достигается при размере ячейки порядка $D = 3l$ и близко к оптимальному при $l < D < 10l$.

Рис. 3. Полное время обработки файла T (слева)
и объем используемой оперативной памяти V (справа)
в зависимости от размера ячейки D

Алгоритмы реализованы на языке C++ в 32-разрядном варианте и испытывались на ПК с процессором Intel Core i7 2700 MHz. Результаты работы алгоритма на различных векторных файлах в сравнении с алгоритмом, использующим дерево квадрантов (Finkel, Bentley, 1974), приведены в *таблице.* Из нее видно, что предложенный метод работает значительно быстрее (приблизительно на один порядок величины), особенно при больших размерах файлов. Однако при этом для работы метода требуется больше оперативной памяти.

Таблица. Сравнение алгоритмов индексации и дерева квадрантов с использованием изображения, содержащего $2{,}5 \cdot 10^6$ точек растра и шести векторных файлов

Векторный файл	***Количество полигонов***	***Алгоритм – дерево квадрантов.***		***Алгоритм – индексация***	
		Время работы, с	*Объем памяти, Мб*	*Время работы. с*	*Объем памяти, Мб*
test1.shp	7951	1,943	0,293	0,128	6,853
test2.shp	18486	3,244	0,663	0,167	6,531
test3.shp	344	0,972	0,014	0,11	3,161
test4.shp	273	2,069	0,011	0,111	2,723
test5.shp	589	3,177	0,02	0,101	2,531
test6.shp	805	2,025	0,03	0,103	1,817

Вывод

Разработан метод, позволяющий ускорить совместную обработку растровых и векторных данных ДЗЗ путем индексации векторного файла. Ключевым условием эффективности индексации является верный выбор размера ячейки. Оптимальное значение – порядка среднего размера полигона – было обосновано теоретически и подтверждено практическими испытаниями. Предложенный метод имеет набор ограничений, за пределами которых он заведомо не является эффективным, однако на реальных данных, отвечающих этим ограничениям, при условии подбора правильного размера ячейки, он дает хорошие результаты, обеспечивая ускорение по сравнению с методом дерева квадрантов.

Работа выполнена при финансовой поддержке Министерства образования и науки РФ (ГК 14.515.11.0003, 14.B35.21.0652) и РФФИ (проекты №№ 13-05-12019\13, 12-01-91162).

Литература

1. *Бондур В.Г., Матвеев И.А., Мурынин А.Б., Трёкин А.Н.* Распознавание выгоревших территорий на мультиспектральных изображениях с адаптируемой маской облачности // Известия Южного федерального университета. Технические науки**.** 2012. Т. 131**.** № 6. С. 153–156.
2. *Бондур В.Г.* Актуальность и необходимость космического мониторинга природных пожаров// Вестник ОНЗ РАН. 2010. Т. 2. NZ11001.
3. *Agoston M.K.* Computer Graphics and Geometric Modeling // Springer. 2004. 920 p.
4. *Guttman A.* R-Trees: A dynamic index structure for spatial searching // Proc. 1984 ACM SIGMOD International Conference on Management of Data. 1984. P. 47–57
5. *Haines E.* Point in polygon strategies // Graphics Gems IV. 1994. PP. 24–46
6. *Finkel R., Bentley J.L.* Quad trees: A data structure for retrieval on composite keys// Acta Informatica. 1974. No. 4 (1). P. 1–9.

7. *Xiongfei Wen.* A new prompt algorithm for removing the bowtie effect of MODIS L1B data // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B3b. Beijing, 2008.

A method of computational optimization for matching vector and raster remote sensing data

V.G. Bondur [1], A.B. Murynin [1, 2], I.A. Matveev [1, 2], A.N. Trekin [1, 3], I.A. Yudin [1]

[1] *"AEROCOSMOS" Institute for Scientific Research of Aerospace Monitoring, Moscow, Russia*
E-mail: amur@ccas.ru
[2] *Dorodnicyn Computing Centre of Russian Academy of Sciences, Moscow, Russia*
E-mail: matveev@ccas.ru
[3] *Moscow Institute of Physics and Technology (State University)*
Dolgoprudny, Moscow rgn., Russia
E-mail: alexey.trekin@gmail.com

Various tasks of remote sensing require combined processing of raster and vector data of the territory. Thus the problem of matching geo-referenced raster against polygons in geographical coordinates arises. Simple method of spatial indexation of the vector polygons is proposed, which increases the speed of vector data search and facilitates the matching. The algorithm has been implemented and tested with real remote sensing data and its performance was compared to analogue. Algorithmic complexity and the way of setting optimal parameters of indexation are analyzed theoretically and confirmed by experiments.

Keywords: spatial indexing, image processing, remote sensing