УДК 004.92

Способы формирования топологии полигональной модели

Юрченко Александр Алексеевич yuaa21r327@student.bmstu.ru
Бурков Павел Викторович burkovpv@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

Полигональные модели в современных САПР-решениях чаще всего представляют собой лишь неструктурированные списки полигонов, что затрудняет выполнение анализа, булевых операций и поиска пересечений. В работе формулируется задача построения топологического описания таких моделей и сопоставляются четыре подхода: полный перебор, граф инцидентности «полигон — ребро», октантные деревья и вокселизация пространства. Оценки теоретической сложности и количества итераций иллюстрируются на примере задачи нахождения линии пересечения двух тел порядка 10^4–$2 \cdot 10^4$ треугольников.

Ключевые слова: *полигональное моделирование, топология сетки, граф инцидентности, октантное дерево, вокселизация, булевы операции*

Введение. Полигональные поверхности стали основным форматом представления геометрии в задачах обратного инжиниринга, медицинской визуализации и цифрового производства [1]. Поток данных от 3D-сканеров формирует модели, содержащие десятки и сотни тысяч треугольников. Однако в большинстве геометрических ядер полигональный объект хранит только массивы координат, индексов и нормалей, не описывая явно связи между вершинами и полигонами.

Отсутствие топологии превращает даже элементарные операции в трудоемкий перебор: чтобы найти пересечение двух тел, необходимо проверять каждый полигон первого тела с каждым полигоном второго. При типичных объемах порядка 10^4 полигонов это несколько миллиардов итераций.

Цель работы — систематизировать известные способы упорядочивания полигональных данных и количественно сравнить их на примере задачи нахождения линии пересечения двух тел.

Полный перебор. Данный вариант заключается в том, что никакой дополнительной структуры, связывающей полигоны между собой, не создается. Единственное, что имеется в распоряжении алгоритма, это два линейных массива (контейнера) треугольников.

Пусть требуется найти линию пересечения двух тел или просто отметить все пары пересекающихся треугольников. Алгоритм сводится к проверке каждого треугольника первого тела с каждым треугольником второго [2].

Данный алгоритм не требует дополнительных затрат по памяти, помимо хранения массивов полигонов. Сложность данного алгоритма $O(N1 \cdot N2)$, где $N1$ — количество полигонов первого тела; $N2$ — количество полигонов второго тела.

Таким образом, простой перебор остается полезным лишь как промежуточный этап, например, для верификации или при N <1000, тогда как для инженерных САПР с сетками 10^3–10^4 полигонов необходимо переходить к топологическим ускорителям, описанным в последующих разделах.

Графовое представление топологии. Наиболее простой и одновременно действенный способ придать неструктурированному набору треугольников топологическую осмысленность — построить граф, в котором полигоны играют роль вершин, а ребра графа — соответствующие общие ребра полигонов в модели. Такая схема сразу превращает сетку в объект с явной связностью: поиск соседей, обход контуров, маркировка областей и другие топологические операции сводятся к стандартным алгоритмам обхода графа.

Для одной и той же математической модели существует множество форм хранения: от примитивной матрицы смежности и перечислений множеств до изощренных структур с указателями [3, 4]. Чтобы выбрать конкретный способ, нужно понять, какие операции будут выполняться чаще всего: перечисление соседей, вставка/удаление ребер, случайный доступ, рекурсивный обход или массовый параллельный перебор. Разберем все классические схемы программной реализации графа в памяти компьютера, оценим их объем, скорость доступа и удобство для полигональной топологии для задач типа построения линии пересечения.

Перечисление множеств. При таком способе каждая пара смежных полигонов фиксируется отдельной записью; вся топология хранится как линейный список кортежей «полигон-полигон». Для произвольного ребра найти второй инцидентный полигон просто: достаточно просмотреть список и выбрать запись, содержащую искомое ребро. Однако перечисление всех ребер, прилегающих к заданному полигону, требует полного прохода по всему множеству записей, поскольку связи не сгруппированы по вершинам-полигонам. Сложность такого запроса тем выше, чем больше всего пар в списке, и на больших сетках он становится узким местом.

Матрица смежности. Строится квадратная таблица размером P × P, где P — число полигонов; элемент (i, j) отмечен, если полигоны i и j соседствуют. Память растет как $O(P^2)$, а для типовой сетки в десятки тысяч фасетов таблица получается сильно разреженной: заняты лишь 3–6 элементов в строке, все остальное — нули. Поиск соседа по конкретному ребру начинается с нахождения строки, соответствующей исходному полигону, а затем линейного сканирования этой строки до первого ненулевого элемента, содержащего нужное ребро. После того как ребро найдено, приходится переходить к столбцу матрицы и снова просматривать все элементы строки, чтобы выяснить, какой полигон стоит на противоположной стороне ребра. Таким образом, даже простейший топологический запрос требует двух проходов по длинной, в основном пустой, строке.

Матрица инцидентности. Здесь строки соответствуют полигонам, а столбцы — ребрам; единица ставится, если полигон содержит ребро. Размер таблицы P × E, где E — число ребер, примерно в 1,5 раза больше P. Матрица

становится еще разреженнее: каждая строка содержит ровно три единицы (для треугольной сетки), каждый столбец — две. Для выяснения, какие ребра принадлежат заданному полигону, нужно пройти по всей его строке и собрать номера ненулевых столбцов. Чтобы найти смежный полигон, приходится выбрать нужный столбец ребра и просмотреть все строки, где этот столбец отмечен единицей, выбирая вторую попавшуюся запись. Оба действия выполняются за линейное время от размеров матрицы, что делает процедуру медленной при больших P и E.

Список вершин и список ребер. Перечисленные выше схемы строят единую, но громоздкую таблицу и не предоставляют быстрого доступа ни к соседям полигона, ни к соседям ребра. Альтернатива — отказаться от матрицы в пользу объектно-ориентированного списка. Каждая вершина-полигон хранится как отдельная структура, содержащая атрибуты полигона и указатель на голову связного списка своих инцидентных ребер. Каждое ребро, в свою очередь, является структурой, в которой есть ссылка на вторую инцидентную вершину-полигон, ссылка на следующее ребро в том же списке, а также собственные геометрические или топологические атрибуты (длина, флаг границы, указатель на исходный объект ядра и т. д.).

Такая организация превращает поиск соседей в операцию постоянной сложности: переход по указателю от полигона к списку ребер и выбор нужного ребра выполняется за O(1), а переход к противоположному полигону — еще за O(1). Избыточные затраты памяти на хранение указателей и служебных полей компенсируются тем, что не нужно выделять огромные разреженные массивы: объем памяти растет линейно O(P+E) и не зависит от квадрата числа элементов.

Все описанные графовые конструкции задают только внутреннюю связность одного полигонального тела: для каждого треугольника они мгновенно выдают соседей, с которыми он делит общее ребро. Этого достаточно, чтобы после обнаружения первой пары пересекающихся полигонов запустить последовательный обход графа: переходя от текущего полигона к смежному через инцидентное ребро, алгоритм вытягивает цепочку отрезков до замыкания пространственной линии пересечения.

Нахождение каждой новой точки пересечения начинается с полного последовательного обхода всех возможных пар полигонов: берем очередной треугольник из первого тела и по порядку проверяем его с каждым треугольником второго тела, пока не обнаружим пересечение. Никаких предварительных фильтров нет, поэтому до первого «попадания» алгоритм затрачивает $O(N \cdot N)$ операций.

Как только пересечение найдено, запускается графовый обход. От стартовой пары алгоритм переходит к соседним треугольникам через общие ребра и последовательно строит всю замкнутую линию пересечения. Во время обхода каждый треугольник, уже вошедший в контур, помечается специальным флагом. Благодаря этому при дальнейшем полном переборе пары, где хотя бы один полигон помечен, пропускаются. Таким образом, последовательный об-

ход графа не уменьшает начальный объем перебора, но ускоряет последующие итерации, исключая из проверки уже использованные полигоны.

Октантное дерево представляет собой иерархическую пространственную структуру, обеспечивающую рекурсивное разбиение ограничивающего куба модели на восемь равных октантов [5, 6]. Процесс продолжается до тех пор, пока в листовом узле не окажется не более заданного числа полигонов либо не будет достигнута предельная глубина разбиения. В результате формируется сбалансированное дерево, в корне которого находится габаритный куб, охватывающий всю геометрию, а листьями служат минимальные кубы-ячейки, содержащие отдельные полигоны исходного тела.

Для каждого треугольника вычисляется октантный адрес — последовательность индексов 0…7, кодирующая путь от корня до соответствующего листа. Соседние по ребру полигоны с высокой вероятностью попадут в один и тот же либо в смежные листья, что отражает топологию полигонального тела. Таким образом, неструктурированный список треугольников преобразуется в упорядоченный набор кластеров, а топологическая связность «общее ребро» сводится к простой проверке принадлежности одному листу.

При поиске линии пересечения двух тел достаточно построить два независимых дерева, соответствующие двум полигональным телам, и выполнить их синхронный обход. На каждом уровне проверяется пересечение пар листьев двух октантных деревьев, и только пересекающиеся ветви развиваются дальше. В итоге формируется компактный список кандидатных листьев, внутри которых уже выполняется точечная проверка треугольников стандартными алгоритмами. Такой подход уменьшает число потенциальных пар пересекающихся полигонов.

Сложность синхронного обхода двух октантных деревьев определяется не суммарным числом треугольников N1 и N2, а объемом их взаимного проникновения в пространстве. В худшем случае, когда геометрии пересекаются «плотно» и деревья почти полностью налегают друг на друга, обход посещает O(min(H1,H2)) узлов, где H1, H2 — высоты деревьев (обычно $H \approx \log N$). На практике же большая часть ветвей отсекается на ранних уровнях. Следовательно, асимптотика синхронного спуска остается близкой к $O(\log N1 \cdot \log N2)$.

Рассмотрим работу данного алгоритма на плоскости. В данном случае октантное дерево принимает частный вид квадродерева и имеет 4 листа, а полигональные тела является некоторым триангулированными фигурами на плоскости, представленными на рис. 1. Пронумеруем каждый полигон заданных фигур. Для представленных фигур квадродеревья будут иметь вид, представленный на рис. 2.

Для двух квадродеревьев выполняется итерационная проверка на пересечение листьев. Так, после первой итерации будет отмечено, что главные узлы двух деревьев пересекаются. Следовательно, кандидатами на пересечение будут массив полигонов B и массив полигонов R. Количество итераций будет 9×10=90. После второй итерации глубина проверки увеличивается. Проверяются на пересечения листья B1-B4 и R1-R4. В данном случае пересекаться

будут листья R2 и B4. Следовательно, в качестве кандидатов на пересечение будут следующие массивы: B4 и R, R2 и B, B4 и R1. Количество итераций в таком случае будет 2 × 9 + 2 × 10 + 2 + 2 = 42. Суммарное количество проверок на пересечения будет 132. Простым перебором будет выполнено 14 × × 11 = 154 проверок на пересечения. Данный пример показывает уменьшение числа итераций.

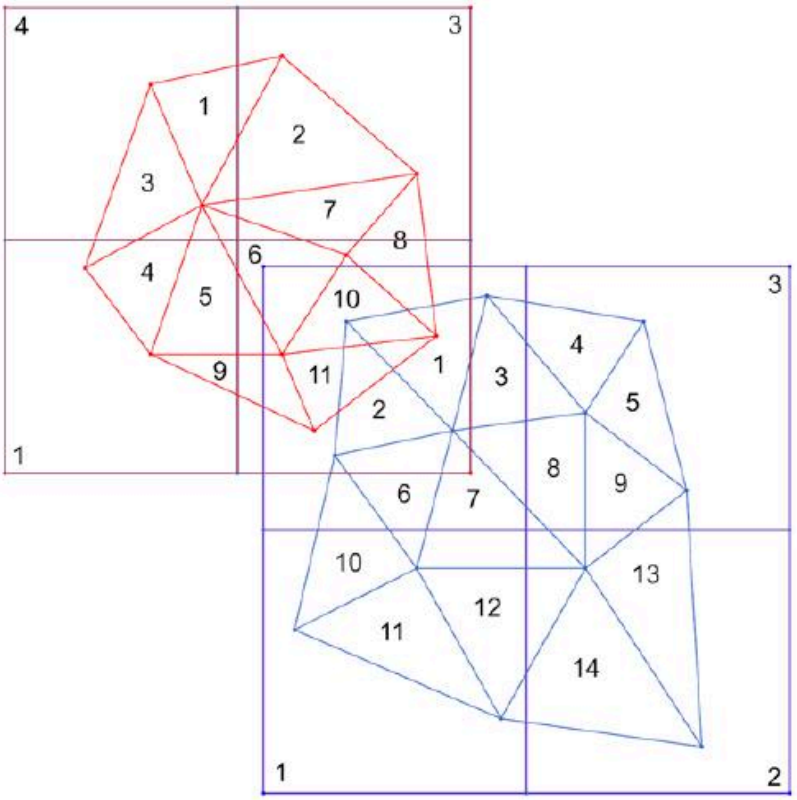

Рис. 1. Исходные фигуры, разбитые на квадродеревья

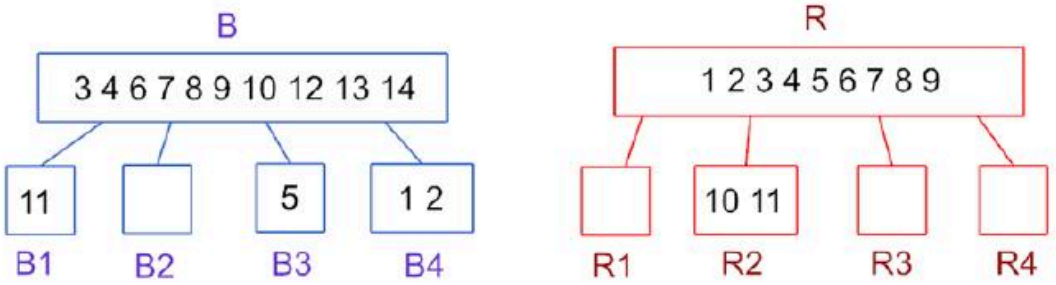

Рис. 2. Иерархическое представление квадродеревьев для исходных фигур

Столь малый прирост в эффективности алгоритма обусловлен малой глубиной деревьев, а также изначально небольшим числом полигонов в рассматриваемых фигурах. С ростом числа полигонов будет увеличиваться и эффективность данного подхода.

Вокселизация пространства. Для ускорения пересечения полигональных тел альтернативой иерархическим структурам служит вокселизация — покрытие модели регулярной сеткой кубических ячеек (вокселей) с ребром δ [5, 6]. Каждый воксель, через который проходит хотя бы один полигон, считается «занятым» и хранит список индексов пересекающих его треугольников. Таким образом, пространство представляется трехмерной растровой структурой с возможно разреженным заполнением.

Процесс начинается с растеризации ограничивающего параллелепипеда (AABB) каждого треугольника: по трем осям определяются диапазоны индексов вокселей, попадающих в AABB, после чего строится список потенциально пересекаемых ячеек. Для каждой из них выполняется точечный тест на пересечение треугольника с кубом δ^3. В худшем случае один полигон может пересечь $O((1/\delta)^3)$ вокселей, однако на практике использование хэш-таблицы (sparse hash) сводит затраты к $O(N)$ по времени и памяти, поскольку большая часть объема остается пустой.

При сравнении двух тел A и B достаточно пройтись только по вокселям, одновременно присутствующим в обеих воксельных масках. Для каждого такого «перекрытого» вокселя выполняется полный перебор пар треугольников из соответствующих списков. Если среднее число полигонов на воксель невелико, суммарное количество проверок остается приемлемым.

Плюсы воксельного подхода — абсолютная регулярность структуры, простота реализации и естественная параллелизация. Кроме того, построение и обход выполняются за линейное время по числу полигонов, что упрощает анализ асимптотики. Главные минусы — кубический рост памяти при уменьшении δ. Тем не менее, при правильном выборе δ метод обеспечивает конкурентную производительность и широко используется в задачах поиска коллизий.

Заключение. В статье систематизированы и количественно сравнены основные методы описания топологических структур для полигональных тел: полный перебор, граф инцидентности, октантное дерево и вокселизация пространства. Каждый из них демонстрирует свой баланс между скоростью, памятью и простотой реализации: от эталонного, но медленного перебора до масштабируемых, но требующих тонкой настройки иерархических или растровых схем.

В качестве перспективного направления развития можно выделить создание гибридной топологической структуры, объединяющей достоинства нескольких подходов. Октантное дерево может выполнять быстрый первичный отсев кандидатов на пересечение, в то время как в листьях дерева может быть размещен локальный граф инцидентности, обеспечивающий мгновенный обход соседей и построение замкнутых контуров пересечения. Такая схема сохранит линейную асимптотику построения и высокую скорость фильтрации, при этом избавит от избыточной траты памяти, характерной для мелкой вокселизации, и даст гибкость при локальных изменениях модели.

Литература

[1] Норенков И.П. *Основы автоматизированного проектирования*. Москва, Изд-во МГТУ им. Н.Э. Баумана, 2009, 430 с.

[2] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to Algorithms*. Cambridge, MIT Press, 2022, 1312 p.

[3] Зыков А.А. *Основы теории графов*. Москва, Вузовская книга, 2004, 664 с.

[4] Голованов Н.Н., Ильютко Д.П., Носовский Г.В., Фоменко А.Т. *Компьютерная геометрия: Основы дифференциальной геометрии и топологии. Основные понятия компьютерной геометрии. Геометрическое моделирование*. Москва, Ленанд, 2024, 504 с.

[5] Ericson C. *Real-Time Collision Detection*. Boca Raton, CRC Press, 2005, 632 p.

[6] Schneider P., Eberly D. *Geometric Tools for Computer Graphics*. Amsterdam, Morgan Kaufmann, Elsevier Science, 2003, 1056 p.

Methods of forming the topology of a polygonal model

Yurchenko Aleksandr Alekseevich **yuaa21r327@student.bmstu.ru**

Burkov Paul Viktorovich **burkovpv@bmstu.ru**

BMSTU, Moscow, Russia

Polygonal models in modern CAD solutions most often represent only unstructured lists of polygons, which makes it difficult to perform analysis, Boolean operations, and intersection search. The paper formulates the task of constructing a topological description of such models and compares four approaches: a complete search, a polygon-edge incident graph, octant trees, and voxelization of space. Estimates of the theoretical complexity and the number of iterations are illustrated by the example of the Boolean operation of subtracting two bodies of the order of 10^4–$2 \cdot 10^4$ triangles.

Keywords: *polygonal modeling, mesh topology, incidence graph, octant tree, voxelization, Boolean operations*