Построение согласованной триангуляции для b-rep моделей с параметрическим представлением поверхностей в системе 3DVision

И. Матвеев, Ю. Игнатьев, В. Микушин
Capvidia BVBA NN, Нижний Новгород

В статье описывается алгоритм построения согласованной триангуляции для b-rep моделей в системе 3DVision.

Триангуляция должна аппроксимировать поверхности с заданной точностью. Условие согласованности означает, что должна быть обеспечена единая сетка на наборе прилегающих (сшитых) граней.

В описанном подходе разделены этапы согласования сеток и триангуляции отдельных граней. Границы граней аппроксимируются специальным образом, так чтобы обеспечить согласование на сшитых участках. На основе этой аппроксимации далее строится триангуляция.

Триангуляция грани представляет собой итерационный процесс усовершенствования сетки. В процессе усовершенствования могут использоваться различные критерии, которые дают возможность строить сетки зависящие от функции поверхности (data-dependent) и сетки с контролем формы (shape-dependent), в частности триангуляцию Делоне. Отдельно рассмотрены различные критерии.

После триангулирования граней выполняется слияние граничных узлов сетки лежащих на сшитых участках границ, что приводит к согласованной триангуляции.

В статье уделяется внимание практическим аспектам реализации алгоритма и сложным случаям, таким как вычисление меры ошибки для обеспечения адекватной аппроксимации сложной геометрии, триангуляция внешнего и внутренних контуров для построения начальной сетки и др.

Ключевые слова: согласованная сетка, триангуляция, параметрические поверхности, граничное представление

Введение

В системе 3D/Vision как и в большинстве современных САПР для представления моделей используется b-rep представление.

Построение согласованной триангуляции требуется для визуализации, выполнения рассчета характеристик тел, в геометрических алгоримах. Экспорт сеточной согласованной модели требуется в различных приложениях, например стереолитографии.

На сетку могут накладываться различные требования. Если для визуализации достаточно обеспечить аппроксимацию поверхностей с заданной точностью и минимальным количеством треугольников, то например в конечноэлементных приложениях необходимо контролировать форму треугольников. Реализованный алгоритм

обеспечивает построение нерегулярных согласованных сеток с возможностью влияния на форму треугольников через критерии усовершенствования сетки.

Постановка задачи

В b-rep представлении тело ограничено оболочками, которые в свою очередь ограничены гранями. Грань представляет собой параметрическую поверхность ограниченную в пространстве параметров внешним и набором внутренних контуров. С помощью такого представления можно моделировать как тела так и открытые составные поверхности. Такая гибкость обеспечила широкое применение такого представления моделей.

Смежные грани имеют общие участки границы - будем называть такие участки сшитыми (рис.1). Зазор на таких участках не должен превышать величины e_{model} - это обеспечивается геометрическим моделером. Для построения согласованной триангуляции стыковку сеток на этих участках в одну общую (рис.1).

Грань задается параметрической поверхностью, которая в пространстве параметров ограничена наборами кривых образующих контуры - внешний и внутренние (рис.2).

Требуется построить согласованную триангуляцию оболочки, аппроксимирующую ее с заданной точностью e_{approx}. Под точностью аппроксимации понимается мера отклонения поверхности от сетки. В данной реализации эта величина определяется таким образом:

$$err_{approx} = \max_{грани}(\max_{ребра}(err^{edge}_{approx}))$$

,где ошибка на ребре err^{edge}_{approx} будет определена позднее.

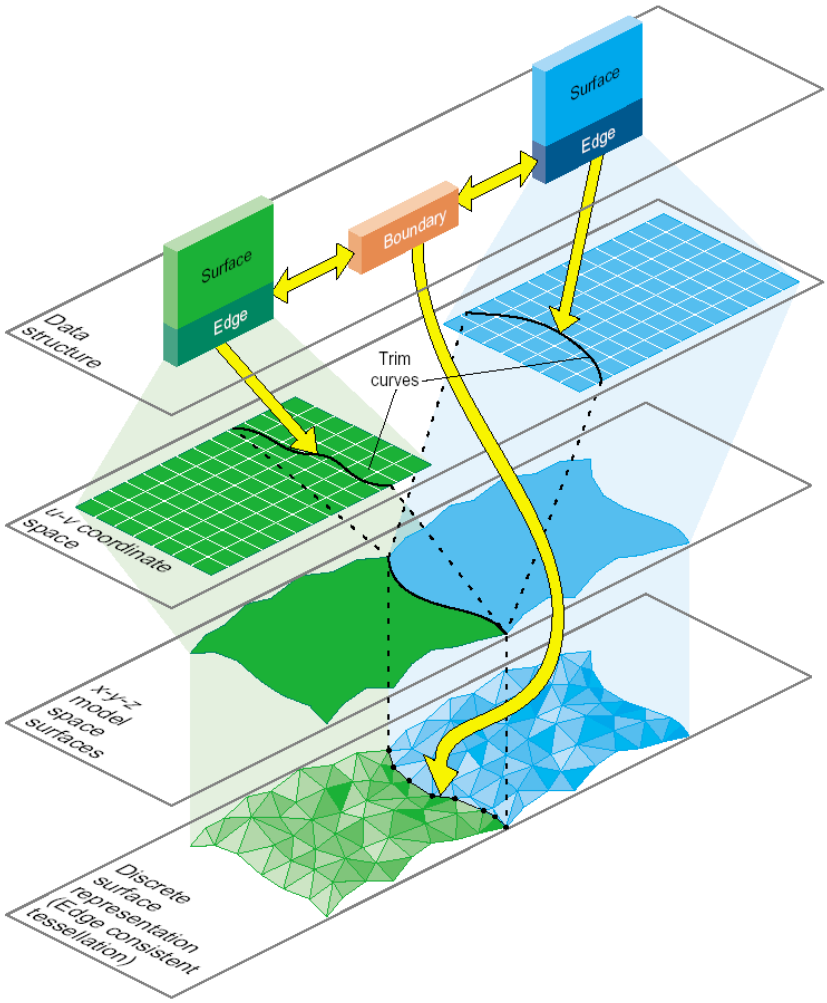


Рис.1 Согласованная триангуляция

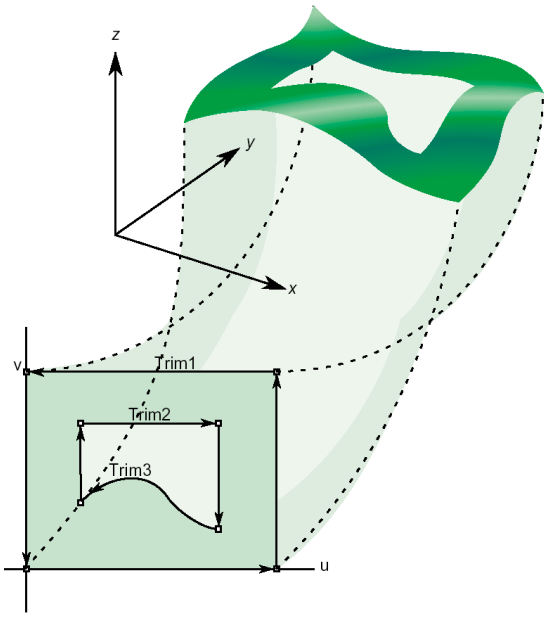


Рис.2 Грань

Идея алгоритма

Аппроксимируем сшитые участки границы так, чтобы для каждой вершины была парная на расстоянии не более e_{model}. Назовем такое построение - согласованной аппроксимацией. Далее триангулируем грани не модифицируея граничные вершины и тем самым сохраним согласованность. Потом остается только выполнить слияние парных вершин на сшитых участках и получить согласованную сетку.

Этапы алгоритма:

- аппроксимация контуров ломаными с точностью e_{approx} и обеспечение согласованности на сшитых участках
- построение начальной триангуляции
- усовершенствование сетки
- слияние сеток на сшитых участках и формирование согласованной сетки

Аппроксимация контура ломаной

Перед триангуляцией граней выполняется аппроксимация их границ (контуров) с точностью e_{approx} [2][4][7]. На основе этого разбиения строится триангуляция. На сшитых участках должна быть обеспечена согласованная аппроксимация.

Алгоритм аппроксимации кривой

Начиная с приближеня кривой отрезком выполняется рекурсивное разбиение до тех пор пока ошибка на всех ребрах не станет меньше e_{approx}:

- за начальное приближение взять ребро 1 (рис.3): $[p_1, p_2]$
- если $err_{approx}^{edge} > \varepsilon_{approx}$, то разбить ребро пополам и повторить этот шаг для двух новых ребер

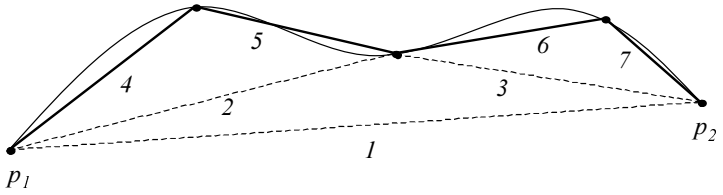


Рис.3 Аппроксимация кривой (4, 5, 6, 7 - результирующие сегменты)

Алгоритм согласованной аппроксимации кривых

При согласованной аппроксимации двух кривых каждая аппроксимационная точка одной кривой должна иметь парную противолежащую точку на другой кривой. Для нахождения парных точек ставится задача минимизации (проецирования). Проецирование происходит на кривую $c_{13}(t)$ лежащую на поверхности, которая задается так:

$$c_{12}(t) = \{u(t), v(t)\}$$
$$s_{23}(u,v) = \{x(u,v), y(u,v)\}$$
$$c_{13}(t) = s_{23}(u,v) \circ c_{12}(t)$$

,где c_{12} - кривая в пространстве параметров поверхности, s_{23} -

поверхность

Алгоритм:

- аппроксимация первой и второй кривой
- выберем базовой кривую у которой получено больше аппроксимационных точек
- нахождение парных точек (проекция с базовой на соседнюю кривую)
- доаппроксимация результата проецирования
- проецирование на базовую кривую точек полученных в результате доаппроксимации

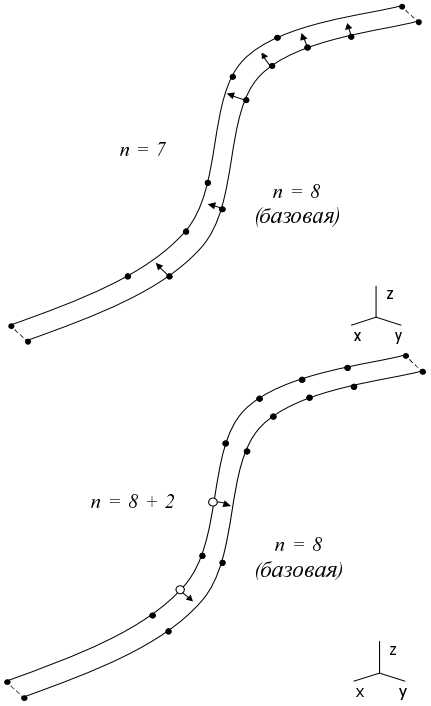


Рис.4 Проецирование базовой аппроксимации (сверху), доаппроксимация и проецирование (снизу)

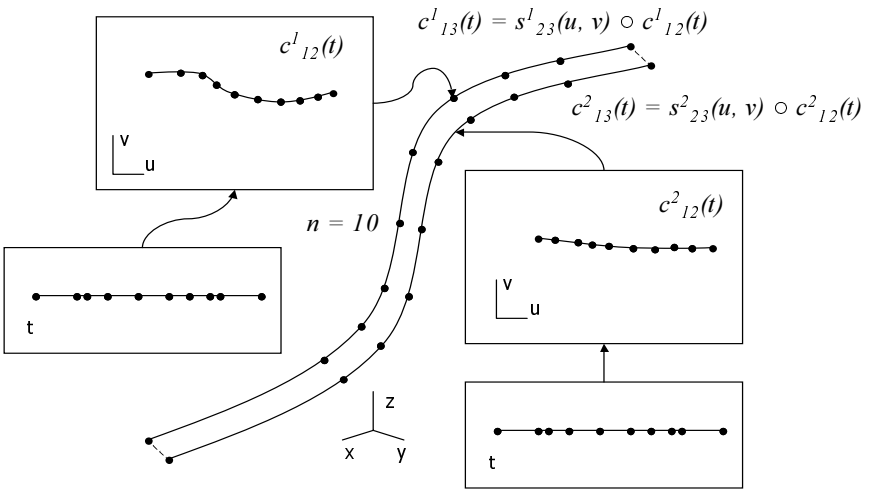


Рис.5 Результат консистентной аппроксимации

Аппроксимация и доаппроксимация выполняется по алгоритму изложенному выше.

Для нахождения парной точки ставится задача минимизации:

$$p^* = \arg\min_{t \in [p_1, p_2]} \left\| p_i - s(u,v) \circ c(t) \right\|$$

,где p^* - неизвестный параметр, p_i - проецируемая точка.

Важное значение имеет хороший выбор начального приближения. Здесь используется информация доступная из имеющейся аппроксимации.

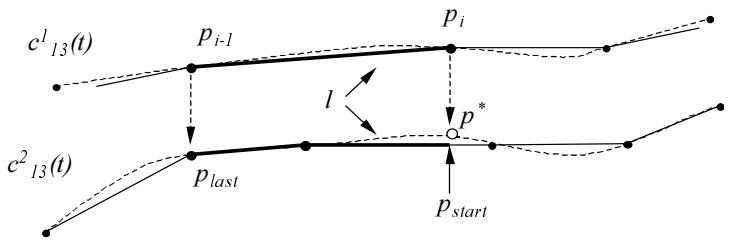


Рис.6 Выбор начального приближения
p_{last} - последняя спроецированная точка, p_i - проецируемая точка, p_{start} - начальное

приближение, p_* - результат проецирования

На рис.6 показан процесс выбора начального приближения p_{start} для проецирования точки p_i. Начальный параметр p_{start} выбираем пропорционально длине кривой рассчитанной в модельном пространстве (3D пространство) по аппроксимации и равной расстоянию *l* между предыдущей p_{j-1} и текущей проецируемой точкой p_i. Отсчет ведем от последней найденной точки p_{last}, чтобы минимизировать ошибку вызванную аппроксимацией. Чем ближе точки между собой тем более линейно ведет себя параметра по отношению к длине кривой, что позволяет использовать линеное приближение. Такого приближения на практике достаточно.

В большинстве практических случаев алгоритм не выполняет шаги после третьего (проецирование базовой аппроксимации), что ускоряет процесс. Однако для поверхностей и кривых с существенно нелинейной параметризацией может потребоваться дополнительная доаппроксимация, которая выполняется на следующих шагах.

Ошибка на ребре

Для рассчета ошибки на ребре рассчитывается невязка в пробных точках ребра [7].

Простейшим видом невязки является отклонение значения функции от приближения на ребре. В случае кривой лежащей на поверхности получаем ситуацию изображенную на рис.7. Пунктиром показаны величины отклонения которые надо проанализировать.

$$err_f^{edge} = \max_p \max(\left\| p^* - s_{23}(u,v) \circ p_{12} \right\|, \left\| p^{**} - s_{23}(u,v) \circ c_{12}(p) \right\|)$$

,где *p* - пробная точка (равномерно распределены в пространстве параметров по ребру), p^* и p^{**} проекции точек на ребро (см. рис.7), p_{12} - средняя точка в пространстве параметров поверхности (пропорционально *p*), p_{13} - средняя точка в модельном пространстве (пропорционально *p*).

Вместо расстояния до проекции можно использовать расстояние до p_{13}.

Кроме проверки отклонения значения функции можно проверять аппроксимацию касательной (рис.8).

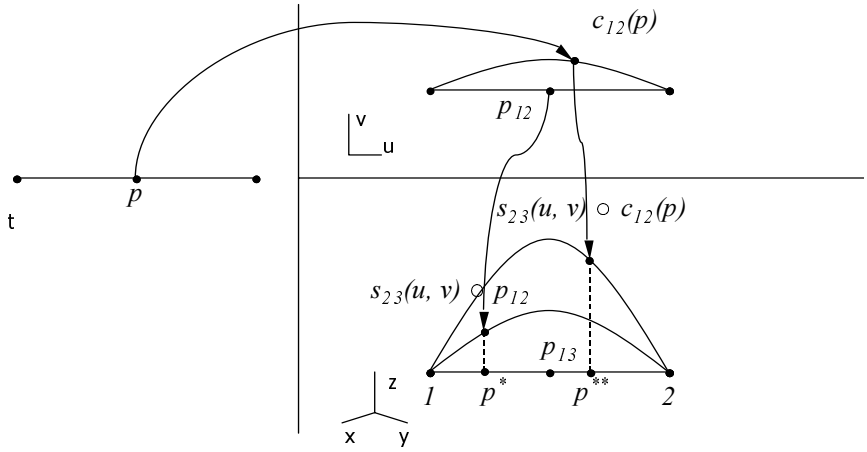


Рис.7 Отклонение значения функции в пробной точке ребра

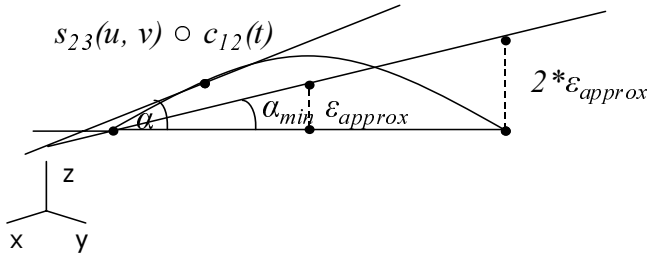


Рис.8 Проверка угла наклона касательной

$$err_{f'}^{edge} = \max_{p} \begin{cases} 0, \alpha < \alpha_{\min} \\ \varepsilon_{approx} + 1 - \cos(\alpha) \end{cases}$$

Формула выдает нарушение условия на касательную, как ошибку в диапазоне $[e_{approx}, 2*e_{approx}]$. Этот прием используется чтобы одинаково интерпретировать отклонение значения функции и касательной. Ааналогичным образом будем поступать с другими типами невязки.

В данном случае проверяется, чтобы касательная была близка к направлению ребра. Мера отклонения зависит от длины ребра и допустимой ошибки (рис.8).

Один из случаев, когда необходима такая проверка показан на ри.9. Пробная точка ребра 1-2 может расположиться таким образом,

что значение функции окажется близко к приближению на ребре. Если не проводить проверку касательной, то такое ребро не будет разбито.

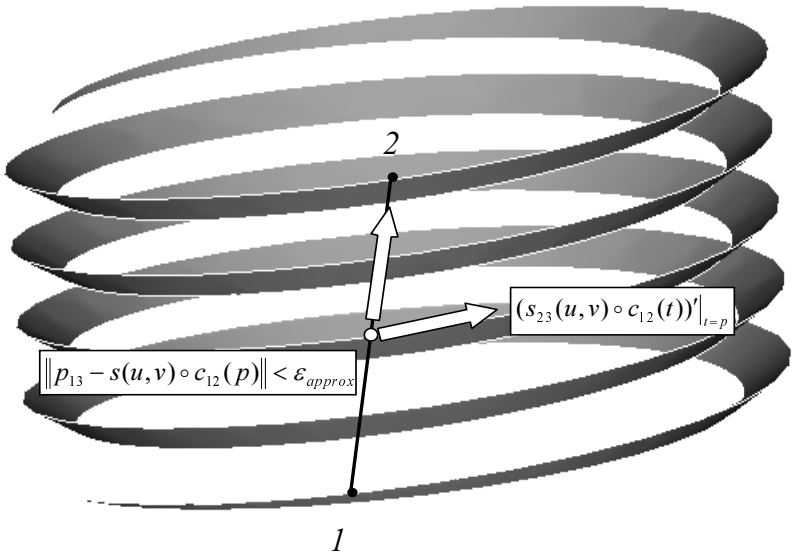


Рис.9 Аппроксимация границы спирали

Рассмотрим еще один вид невязки. Используем значение кривизны в пробных точках. На рис.10 изображено ребро *l* и окружность с радиусом кривизны в точке *p*. Наложим ограничение на соотношение отклонения ребра *d* от дуги окружности к радиусу кривизны. Можно это интерпретировать так: обеспечить заданное количество сегментов для аппроксимации окружности с радиусом кривизны в каждой пробной точке. Если данное ребро не проходит по критерию - они должно быть разбито.

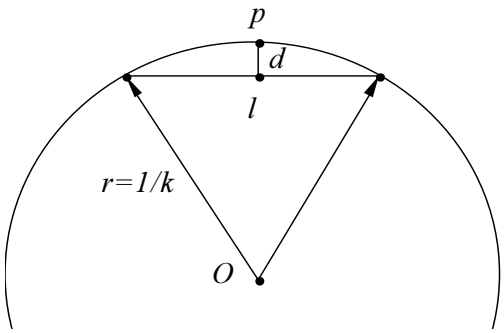


Рис.10 Кривизна в пробной точке *p*

$$k = \max(|k_1|, |k_2|); \quad h = \frac{d}{r} = dk; \quad err_k^{edge} = \max_p \begin{cases} 0, h < h_{\max} \\ \varepsilon_{approx}\left(2 - \frac{h_{\max}}{h}\right) \end{cases}$$

,где *k* - максимальная кривизна, h_{max} задает максимальное соотношение отклонения *d* и радиуса кривизны *r* (используется величина 0.2).

Если необходимо использовать все невязки, то получим:

$$err_{approx}^{edge} = \max(err_f^{edge}, err_{f'}^{edge}, err_k^{edge})$$

Основной является невязка значения функции - err_f^{edge}. Для граней сложной формы (рис.9) нужно дополнительно учитывать $err_{f'}^{edge}$. Невязка err_k^{edge} может быть использована для более строгой проверки точности аппроксимации, но обычно достаточно первых двух оценок и кроме того - она самая длительная по времени рассчета.

Отдельно стоит отметить, что на практике при рассчете ошибки на ребре достаточно брать три пробные точки.

Ошибка на ребре для согласования с триангуляцией

Если аппроксимация границы будет выполнена недостаточно мелкими элементами, то в процессе усовершенствования сетки алгоритм может зациклиться пытаясь улучшить форму треугольников

или ошибку на ребре в окрестности границы. Для того чтобы этого избежать необходимо согласовать функцию вычисления ошибки на ребре с критерием усовершенствования сетки. Поэтому на границе должна использоваться ошибка на ребре более строгая чем внутри области или такая-же если требуется только обеспечить точность.

Один из вариантов решения этой задачи - обеспечить такое разбиение, которое локально удовлетворило бы любой кривой на поверхности проходящей через точку.

Обратимся к рис.10. Предположим, что ребро *l* имеет оптимальную длину в окрестности точки *p* с кривизной *k*, тогда $d = e_{approx}$ и далее:

$$d = \varepsilon_{approx}; \quad l^2 = l_{\min}^2 = 4d(2r - d); \quad err_{kl}^{edge} = \begin{cases} 0, l_{edge} < l_{\min} \\ \varepsilon_{approx}\left(2 - \dfrac{l_{\min}}{l_{edge}}\right) \end{cases}$$

Получили выражение для величины оптимального ребра l_{min}. Эта величина зависит от e_{approx} и максимальной кривизны *k*. Получили условие на длину сегмента для аппрокимации в данной точки с учетом максимальной кривизны поверхности.

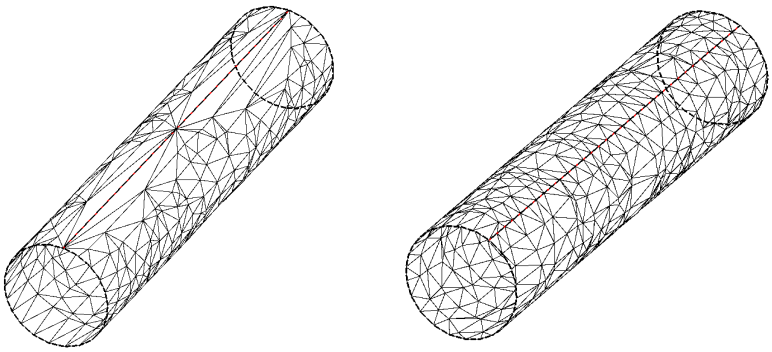


Рис.11 Слева - обычная ошибка на ребре, справа - с контролем размера по кривизне поверхности

Граница грани является прямой (рис. 11), но тем не менее происходит разбиение, т.к. поверхность в этих точках имеет ненулевую кривизну.

Построение начальной триангуляции

Алгоритм “отрезания уха”

Для построения начальной триангуляции на основе аппроксимации контура используется алгоритм “отрезания уха” [1].

На контуре выбирается три последовательно идущие вершины и если на этих вершинах можно построить допустимый треугольник (в который не попадают другие вершины контура), то происходит создание треугольника и из контура удаляется точка. Этот процесс повторяется пока все вершины не будут исчерпаны.

В 3D/Vision реализовано несколько вариантов алгоритма выбора следующей вершины и специальные структуры данных для ускорения проверки кандидатов на построение треугольника. От порядка выбора вершин зависит качество получаемых треугольников.

Вершины с углом <180 градусов могут служить кандидатами для построения треугольника, поэтому для эффективности нужно поддерживать список таких вершин. Для проверки допустимости построения треугольника надо проверить только вершины с углом >180 градусов, отсюда следует, что такие вершины тоже удобно хранить в специальном списке. На практике часто приходится иметь дело с прямоугольными контурами, когда есть большое количество точек с углом 180 градусов. Поэтому чтобы не тратить времени на обработку таких вершин поддержка списков становится особенно актуальной.

Существуют два класса алгоритмов выбора вершины - последовательный выбор и выбор с сортировкой по критерию [1]. В 3D/Vision реализованы оба.

Простой последовательный выбор с фиксированным направлением обхода часто приводит к “веерам” треугольников с малыми углами. В 3D/Vision за основу выбран модифицированный вариант последовательного выбора. После “отрезания уха” происходит поиск следующего кандидата с обходом контура в сторону противоположную той, что использовалась для поиска данной точки. Кроме этого надо пропустить одну точку контура. Такая простая модификация позволяет строить приемлимые (по числу тонких треугольников) сетки в практически важном случае прямоугольного контура. В других случаях этот ведет себя не хуже простого последовательного выбора.

Кроме этого реализовано три варианта алгоритма выбора с сортировкой по критерию, они в ряде случаев дают более качественные сетки, но это дается замедлением работы этой фазы алгоритма примерно на 15-20% [1]. Реализованы следующие критерии:

- максимальный синус угла
  - выбираем треугольники наилучшей формы

- максимальная площадь отсекаемого треугольника
  - выбираем большие треугольники
- максимальное увеличение суммарной площади отсекаемой всеми кандидатами (разница до и после отсечения данного кандидата)
  - выбираем такой треугольник для отсечения, чтобы после его отсечения суммарная площадь была как можно больше, т.е. больше шансов на следующем шаге получить кандидатов с болшой площадью

Каждый из этих критериев ведет себя лучше в том или ином случае и при необходимости можно применять их последовательно для достижения оптимального результата. Применять разные критерии имеет нужно если получаются очень тонкие треугольники и стандартный последовательный алгоритм не справляется. Процент таких случаев крайне мал.

Алгоритм “отрезания уха”:

- строится список вершин <180
- строится список вершин >180
- строится приоритетный список вершин по критерию
- выбрать кандидата последовательным выбором или по критерию
- проверка случая “веера” - специальное окончание алгоритма
- проверка допустимости кандидата на всех вершинах >180, если не допустим, то вернуться к шагу выбора кандидата
- строить треугольник, исключить вершину из контура, обновить списки
- перейти к шагу выбора кандидата

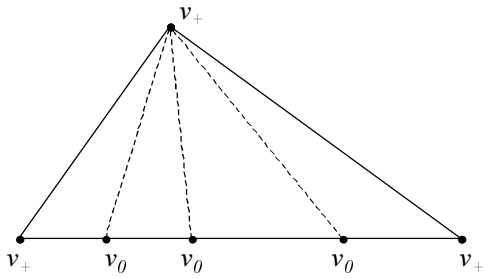


Рис.12 Случай “веера”. v_+ - кандидаты, v_0 - вершины с углом 180 градусов

Отдельно проверяется случай “веера” изображенный на рис.12.

В этом случае имеется три кандидата, остальные вершины с углом 180. Если этот случай не распознавать, то он проходит по критерию допустимости и приводит к тупику.

Алгоритм “отрезания уха” триангулирует один контур, но еще требуется вырезать внутренние контуры. В 3D/Vision реализовано два подхода к решению этой задачи. В первом случае строится один контур получаемый стыковкой всех вместе с помощью “мостов”. Во втором случае точки внутренних контуров добавляются в существующую триангуляцию таким образом, чтобы ребра внутренних контуров присутствовали в триангуляции тем самым восстанавливая нужный контур.

Алгоритм “мостов”

Рассмотрим так называемый алгоритм “мостов” [1]. Алгоритм соединяет внешний и внутренние контуры вырожденными перемычками “мостами”. Далее для такого контура с минимальными изменениями применяется стандартный алгоритм “отрезания уха”.

Алгоритм:

- упорядочить узлы всех контуров по возрастанию x
- упорядочить внутренние контуры по самой левой точке по возрастанию координаты x
- для каждого внутреннего контура (слева направо) выполнить:
  - искать подходящий “мост” от любой точки левее самой левой точки данного внутреннего контура, кандидатом считать отрезок соединяющий выбранную точку и самую левую точку текущего контура
  - каждый кандидат проверяется на пересечение со всеми отрезками лежащими левее крайней точки текущего контура, если пересечений нет, то строится “мост”

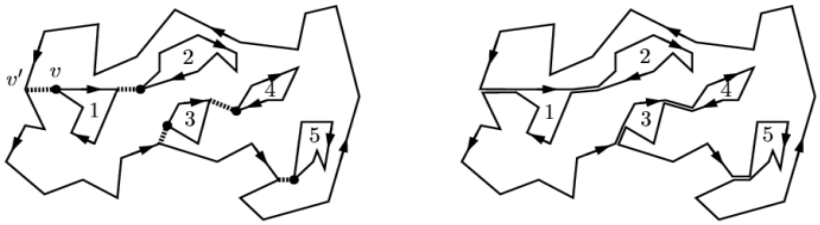


Рис.13 Формирование ”мостов”

Сложность такого алгоритма оценивается как $O(i*n^2)$, где i - число внутренних контуров. Алгоритм прост в реализации и устойчив, однако при большом числе внутренних контуров неэффективен.

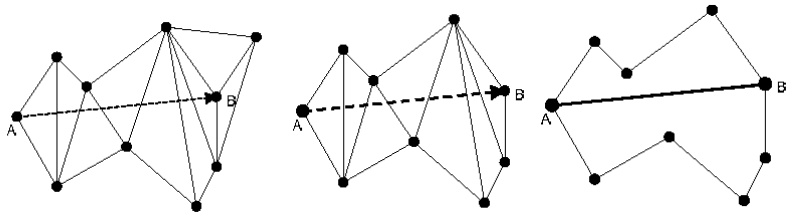
Для адаптации алгоритма “отрезания уха” к контурам полученным после построения “мостов” требуется внести незначительные коррективы. На “мосте” нужно обеспечить связь треугольников с обоих сторон. Для этого в узле добавляется ссылка на парный узел. При построении треугольника и при наличии такой связи необходимо проверить нет ли построенного треугольника с другой стороны “моста” и если есть установить с ним топологическую связь.

На практике могут возникать случаи, когда приходится строить несколько “мостов” исходящих из одной вершины, надо это иметь ввиду при реализации и обеспечить корректную обработку этой ситуации.

Восстановление внутренних контуров

Алгоритм (применяется для каждого внутреннего контура) [5]:

- найти положение первой точки контура в триангуляции
- триангулировать точку соединив с соседними вершинами (в случае попадания на ребро - разрезаем ребро)
- для каждой следующей точки контура:
  - найти положение точки в сетке стартуя с предыдущей
  - триангулировать точку
  - выделить ребра которые пересекают ребро соединяющее текущую и предыдущую точки
  - выделить полигон ограничивающий эти ребра
  - удалить треугольники в этом полигоне
  - построить искомое ребро соединяющее текущую и предыдущую вершину
  - триангулировать две половины рассеченного полигона



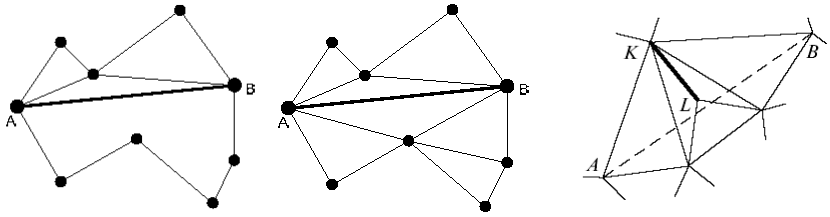


Рис.14 Процесс восстановления ребра

Процесс работы алгоритма для одной точки показан на рис.14.

Существуют случаи когда получаемый полигон содержит ребра обходящиеся дважды (последняя иллюстрация на рис. 14). В этом случае по аналогии с тем что делалось с “мостами” надо обрабатывать связи на этих ребрах.

При последовательном восстановлении вершин контура выполняется условие “прямой видимости” следующей вершины из предыдущей. Поиск положения следующей точки выполняется перешагиваением по триангуляции по направлению луча [6][3] соединяющего одну точку с другой и т.к. это близкие точки, то эта операция выполняется очень быстро. При реализации следует обратить внимание на случаи, когда направляющий вектор пересекает вершину или проходит через ребро касаясь его.

После триангуляции внутренних контуров остается удалить лишние треугольники находящиеся внутри этих контуров. Это делается обходом всех вершин внутренних контуров с удалением прилегающих треугольников с учетом направления обхода контура.

Усовершенствование сетки

На этом этапе сетка модифицируется так, чтобы обеспечить выполнение условия аппроксимации с заданной точностью. Точность аппроксимации определяется через функцию ошибки на ребре сетки. Условием завершения является обеспечение заданной точности на всех ребрах

В процессе усовершенствования [3] в триангуляцию добавляются новые вершины в те места где отклонение сетки от исходной поверхности максимально. После этого в окрестности добавленной вершины выполняется процедура локальной оптимизации [3], которая выполняет локальную перестройку сетки по заданному критерию. Через этот критерий можно влиять на свойства триангуляции.

Алгоритм усовершенствования сетки:

- выполнить глобальную оптимизацию для начального приближения
- сформировать приоритетный список треугольников упорядоченный по ошибке, в список попадают треугольники с ошибкой больше заданной
- выбрать треугольник с максимальной ошибкой, в нем ребро с максимальной ошибкой
- добавить на ребро с максимальной ошибкой вершину [3][7]
- выполнить процедуру локальной оптимизации для треугольников прилегающих к новой вершине, в ходе операции обновлять приоритетный список

Глобальная оптимизация требуется для приведения триангуляции в соответствие с заданным критерием оптимизации. Она реализуется применением локальной оптимизации ко всем вершинам триангуляции до тех пор пока локальная оптимизация дает эффект (осуществляет модификацию сетки).

Локальная оптимизация изначально была предложена для внедрения точек в Делоне триангуляцию, но может использоваться и с другими критериями [3].

Алгоритм локальной оптимизации:

- для каждого треугольника прилегающего к вершине:
  - если переброска невозможна, то выход
  - если по критерию оптимизации ребро *DB* лучше ребра *AC* (рис.15), то выход
  - выполнить переброску
  - выполнить локальную оптимизацию для ребер *DC*, *BC*

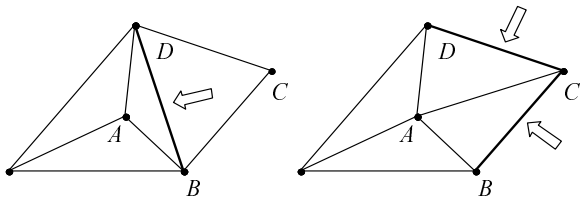


Рис.15 Локальная оптимизация после добавления вершины *A*
(стрелками помечены ребра на которых делается проверка переброски)

Процесс оптимизации сходящийся. Введем вектор $Q = \{E_1, E_2, \ldots, E_n\}$ показывающий качество триангуляции, компонентами которого

являются упорядоченные по убыванию ошибки на треугольниках. Введем операцию сравнения таких векторов в лексикографическом порядке, отсюда понятно, что *Q'*, полученный в результате локальной оптимизации будет всегда меньше *Q*, поэтому процесс переброски сходится.

Критерий минимизации ошибки на ребре

Простейшим критерием оптимизации является минимизация ошибки на ребре. Такой критерий приводит к построению триангуляции без контроля формы (data-dependent). Треугольников при этом получается немного. Условие переброски:

$$err_{approx}^{edge}(DB) > err_{approx}^{edge}(AC)$$

Критерий минимизации максимального угла в модельном пространстве

Эквивалентом круговому критерию для триангуляции Делоне является максимизация минимального угла. Аналогичный критерий можно применить в модельном пространстве. Условие переброски:

$$\min\left(\min_{\alpha} ABC, \min_{\alpha} ACD\right) > \min\left(\min_{\alpha} ABD, \min_{\alpha} DBC\right)$$

Прямое использование этого критерия иногда приводит к плохому результату. Триангуляция начального приближения плохо приближает геометрию и возможны случаи когда формируются ребра выгодные по этому критерию, но плохо ориентированные относительно поверхности. Выход из положения дает следующий критерий.

Комбинированный критерий

Комбинированный критерий сначала обеспечивает точность аппроксимации близкую к заданной, и далее уже применяется критерий минимизации угла. Условие переброски:

$$\begin{cases} err_{approx}^{edge}(DB) < \varepsilon_{approx} k \rightarrow err_{approx}^{edge}(DB) < err_{approx}^{edge}(AC) \\ \min\left(\min_{\alpha} ABC, \min_{\alpha} ACD\right) > \min\left(\min_{\alpha} ABD, \min_{\alpha} DBC\right) \end{cases}$$

,здесь *k* - коэффициент приближения к требуемой точности

(можно использовать *k=3*)

Критерий с использованием I квадратичной формы поверхности

С помощью I квадратичной формы поверхности можно описать дифференциал длины дуги кривой лежащей на поверхности:

$$r(t) = r\left[u(t), v(t)\right]$$
$$ds^2 = \left|dr\right|^2 = E(u,v)du^2 + 2F(u,v)dudv + G(u,v)dv^2$$

Рассмотрим треугольник в пространстве параметров. Будем искать точку в пространстве параметров, такую, чтобы в модельном пространстве она была равноудалена от вершин треугольника. Запишем уравнение прямой проходящей через эту точку и получим:

$$\begin{cases} u(t) = C_u t + (1-t)u \\ v(t) = C_v t + (1-t)v \end{cases} \rightarrow \begin{cases} u'(t) = C_u - u \\ v'(t) = C_v - v \end{cases}$$
$$s^2 = E(C_u - u)^2 + 2F(C_u - u)(C_v - v) + G(C_v - v)^2$$

Подставляя в выражение для s^2 координаты трех вершин треугольника получим систему трех уравнений для нахождения s^*, C_u и C_v. Зная C получаем аналог кругового критерия с использованием выражения для квадрата расстояния:

$s(u,v) < s^*$

Критерий минимизации максимального угла в пространстве параметров

Этот критерий эквивалентен круговому критерию для построения триангуляции Делоне.

Далее приведены результаты применения разных критериев.

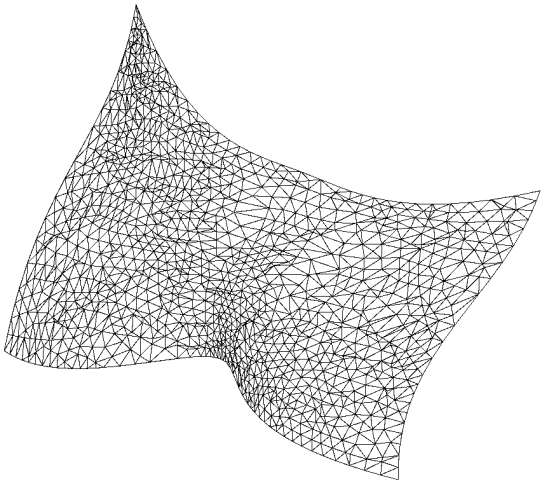


Рис.16 Комбинированый критерий (2481 тр.)

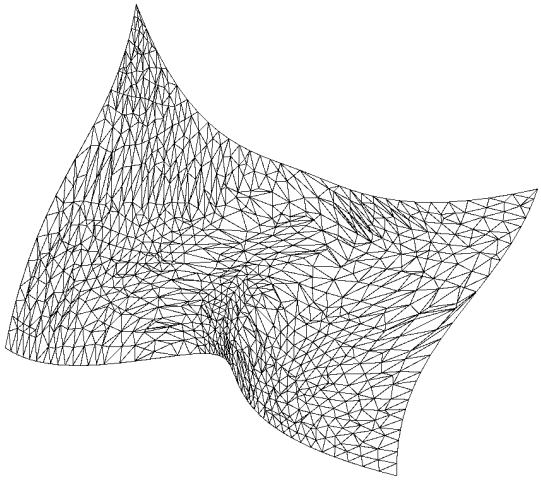


Рис.17 Критерий минимизации ошибки (1923 тр.)

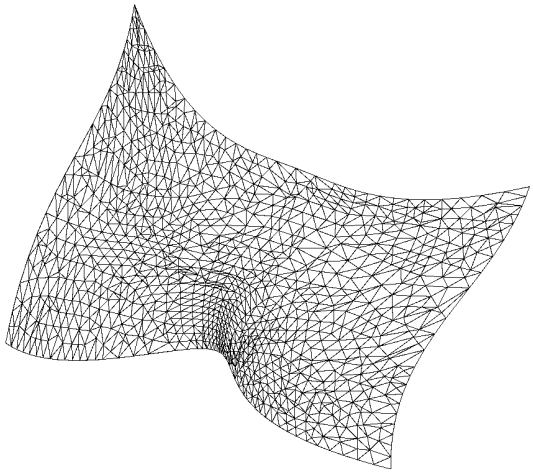


Рис.18 Триангуляция Делоне (2215 тр.)

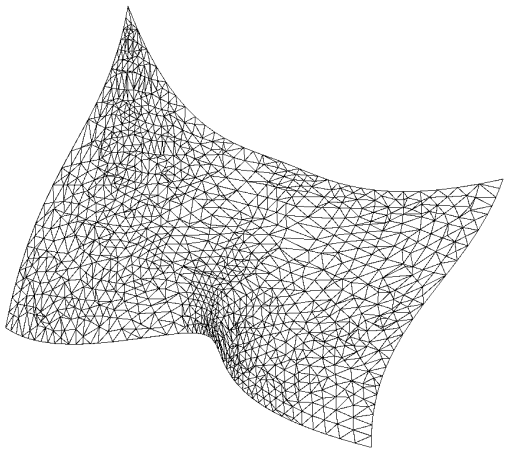


Рис.19 Критерий с I квадр. формой (2483 тр.)

Построение согласованной триангуляции

В результате триангуляции граней получается сетка которая на сшитых участках согласована с соседними гранями с точностью e_{model}.

Для построения согласованной сетки надо устранить эти зазоры. Это делается слиянием соотвествующих вершин (рис.20). Вершины находящиеся на стыке нескольких граней обрабатываются не попарно, а выбирается одна вершина из всех и она используется для всех граней как показано на рис.20.

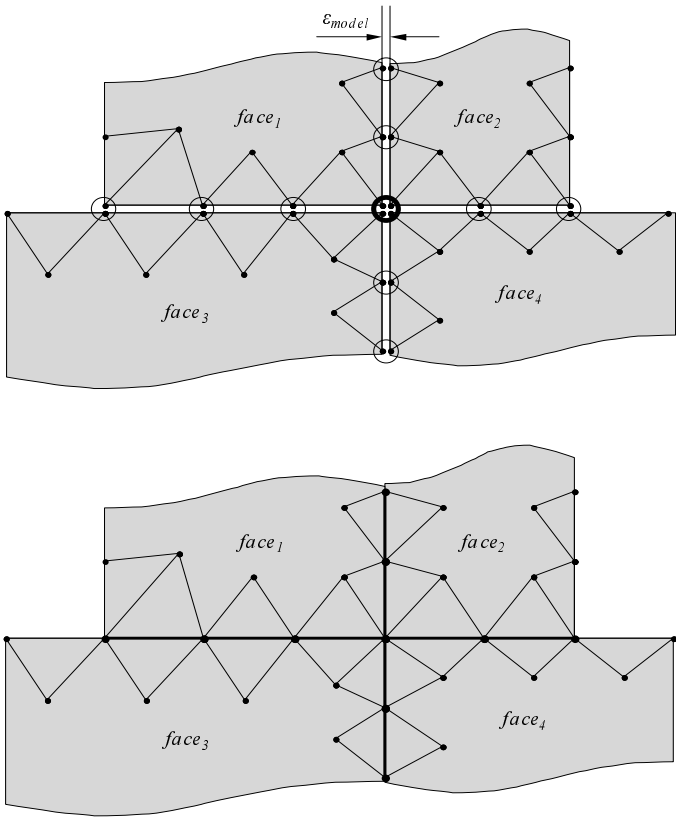


Рис.20 Построение согласованной сетки

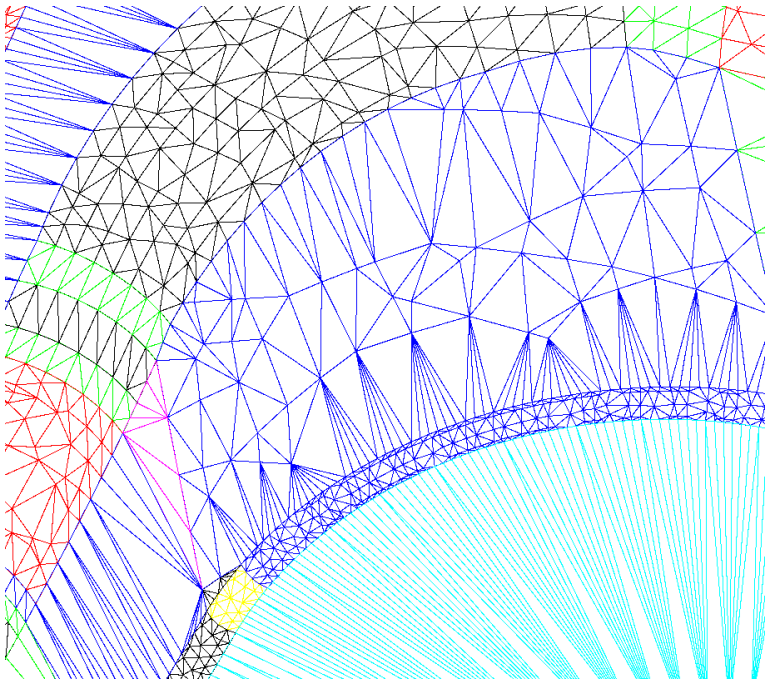


Рис.21 Пример согласованной триангуляции (разным цветом помечены разные грани)

Заключение

Предложен подход для построения нерегулярной согласованной триангуляции b-rep моделей. Согласование сеток достигается согласованнием аппроксимации сшитых границ граней. Описан алгоритм аппроксимации использующий ошибку на ребре для уточнения апроксимации. Рассмотрено несколько способов рассчета ошибки. Учет кривизны поверхности в граничных точках позволяет наложить ограничение на размер аппроксимирующих ребер, чтобы избежать построения вытянутых треугольников. При построении начальной триангуляции использован алгоритм “отрезания уха” с последовательным и приоритетным выбором кандидатов. Модифицированный последовательный выбор позволяет в ряде практических случаев обеспечить более качественную триангуляцию без дополнительных вычислительных затрат. На случай возникновения проблем с тонкими треугольниками при последовательном выборе имеется три варианта выбора кандидата по приоритету.

Усовершенствование сетки вводит ввершины в триангуляцию в районе максимальной ошибки аппроксимации. После введения вершины применяется процедура локальной оптимизации. Рассмотрены несколько критериев оптимизации для построения data-dependent и shape-dependent сеток. На последнем этапе зазоры на сшитых участках устраняются слиянием парных вершин на границе граней.

В будущем требуется обеспечить плавный градиент плотности, контроль размера и углов треугольников. Для этого возможно потребуется изменение положения вершин и новые критерии переброски.

Литература

[1] Martin Held “FIST: Fast Industrial-Strength Triangulation of polygons” // Algorithmica, vol.30, #4, pages 563-596, 2001

[2] H.L. de Cougny “Surface meshing using vertex insertion” // In Proceedings of the 5th International Meshing Roundtable, pages 243-256, 1996

[3] Michael Garland and Paul S. Heckbert “Fast polygonal approximation of terrains and height fields” // CMU-CS-95-181, 1995 sept.

[4] Hao Chen, Jonathan Bishop “Delaunay triangulation for curved surfaces”

[5] А.В. Скворцов “Алгоритмы построения триангуляции с ограничениями” // Вычислительные методы и программирование, том 3, 2002

[6] А.В. Скворцов “Обзор алгоритмов построения триангуляции Делоне” // Вычислительные методы и программирование, том 3, 2002

[7] Reinhard Klein, Wolfgang Straber “”Mesh generation from boundary models with parametric face representation” // Third Symposium on Solid Modeling and Applications, pages 431-440, 1995

[8] Lyuba Alboul, Gertjan Kloosterman, Cornelis Traas, Ruud van Damme “Best data-dependent triangulations”