СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Возникнув в восьмидесятых годах двадцатого века, компьютерное моделирование ткани бурно развивается по сей день. За годы было предложено множество подходов для решения этой задачи. До недавнего времени, наиболее популярными методами симуляции в реальном времени было представление ткани в виде системы масс и пружин, а также способы, основанные на методе конечных элементов. Каждый из этих подходов имеет свои преимущества и недостатки, но главной их проблемой является сложность разделения задачи между несколькими потоками.

С началом массового выпуска многоядерных процессоров возникла потребность в новом подходе, позволяющем масштабировать вычисления. В 2006 году был создан алгоритм Position Based Dynamics, позволивший получить приемлемое качество симуляции при высокой скорости. Затем, в 2016 году теми же авторами было предложено улучшение оригинальной модели, получившее название Extended Position Based Dynamics. XPBD значительно повысил точность расчетов. Что еще важнее, XPBD, как и PBD позволил разбить вычисления на параллельные потоки, работа которых не зависит друг от друга. Метод обрел популярность, и на протяжении последних лет XPBD является наиболее распространенным способом симуляции ткани в реальном времени.

Отметим, что ограниченное применение нашел также метод, основанный на моделировании взаимодействия микроволокон ткани. Этот способ обеспечивает высокую точность, однако современные компьютеры не все еще недостаточно быстры. Повторимся, что в рамках этой работы рассматриваются лишь подходы, способные обеспечить симуляцию в реальном времени.

В данной работе изучен метод XPBD, а также один и способов распределения вычислений между несколькими ядрами процессора. Кроме того, реализована поддержка коллизий с твердыми телами и трение. Практическим результатом работы является программа, позволяющая производить симуляцию в реальном времени.

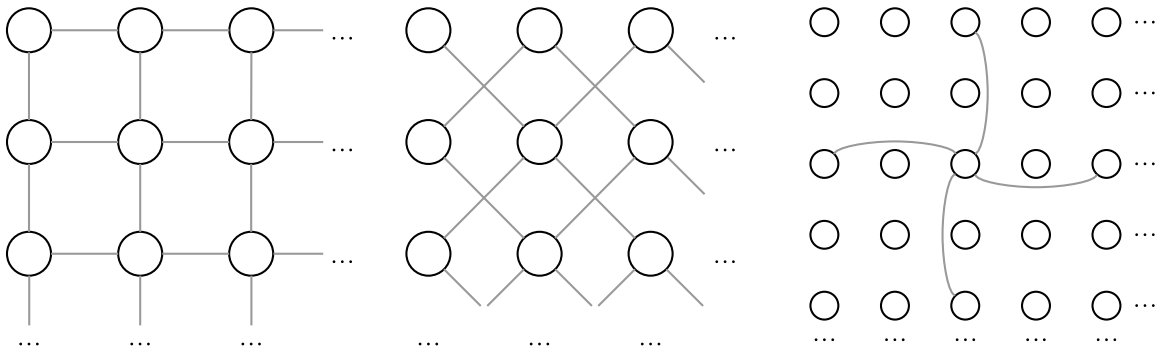
Согласно ПРИЛОЖЕНИЮ А, задачи данной работы таковы:

- обзор литературы по теме работы;
- изучение метода BPD;
- изучение метода XPBD;
- изучение ограничений, применяемых для симуляции ткани;
- поиск способа разделения задачи между несколькими потоками;
- изучение алгоритмов поиска и обработки коллизий.

1 Обзор популярных методов симуляции ткани

1.1 Представление ткани в виде системы масс и пружин

Данный метод является самым простым способом симуляции ткани. Материал представляется в виде множества материальных точек, имеющих определенный вес и соединенных упругими стержнями [1]. В простейшем случае модель содержит лишь пружины, ограничивающие растяжение ткани вдоль двух основных направлений, но возможно также введение диагональных элементов, стесняющих движение полотна в плоскости и дополнительных стержней, препятствующих изгибу. Пружины различных типов изображены на Рисунке 1.



а) б) в)

Рисунок 1 – Система масс и пружин

Таким образом, на точку (i, j) действуют следующие силы:

- внутренние силы ткани, то есть силы, создаваемые стержнями, соединенными с текущей вершиной,

- внешние силы,

- иногда учитывают также вязкое трение.

На очередном шаге симуляции для каждой из вершин вычисляется суммарный вектор сил $\bar{F}_{i,j}$ и тем или иным способом интегрируется уравнение её движения. Чаще всего пользуются явным методом Эйлера, простым или модифицированным методами Верле, а также методом Рунге-Кутты [2].

При своей простоте, представление ткани в виде системы масс и пружин имеет ряд недостатков. Во-первых, физическая модель, принятая в этом подходе, хоть и отражает силы, действующие в реальных материалах, но не имеет привязки к конкретным физическим размерностям. Например, нельзя задать упругость на растяжение в Паскалях.

Вместо этого можно лишь попытаться подобрать жесткость пружин так, чтобы визуально результат походил на ту или иную ткань.

Во-вторых, данный метод чувствителен шагу времени при интегрировании. При более мелком шаге общее качество результата повышается, и полотно становится более упругим. При большом шаге материал напротив начинает неестественно растягиваться, визуально походя на резину. Такое поведение ткани показано на Рисунке 2.

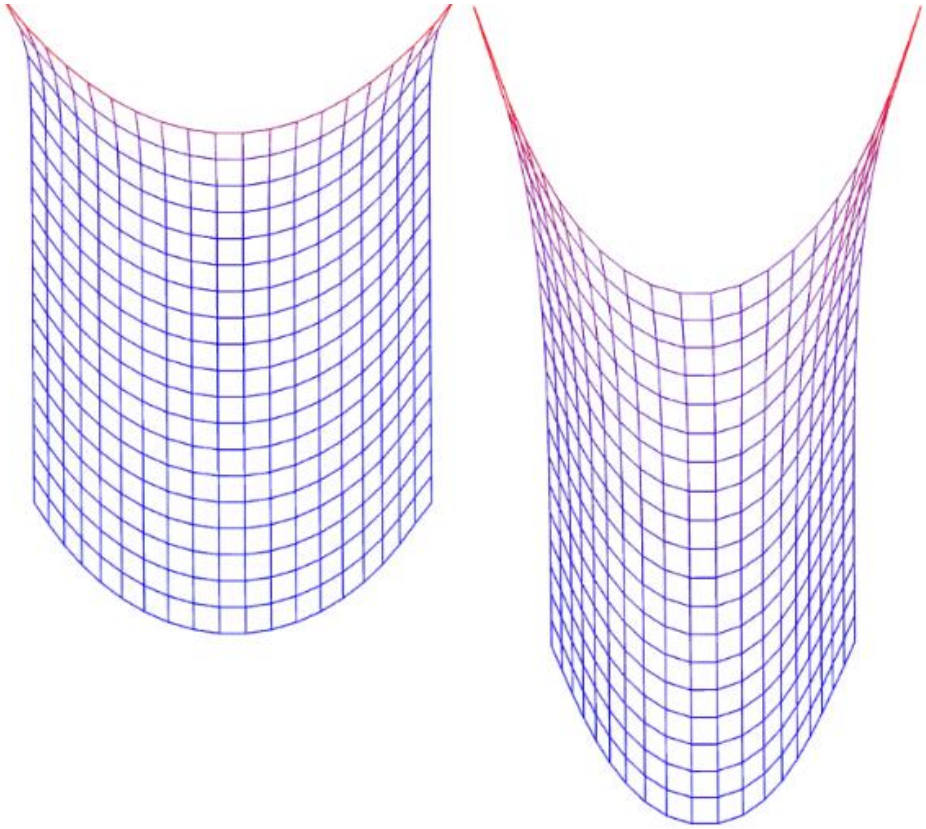


Рисунок 2 – Симуляция ткани интегрированием Верле 600 итераций и 150 итераций

Таким образом, при представлении материала в виде системы масс и пружин приходится «угадывать» жесткость стержней, основываясь на внешнем сходстве поведения моделируемой ткани с реальной, а также следить за тем, чтобы временной шаг был стабилен, ведь его вариация повлечет за собой заметное изменение свойств полотна. Эти недостатки можно попытаться нивелировать, обходя вершины ткани по несколько раз на каждом шаге симуляции, и уточняя их координаты [3]. Это улучшает положение, однако принципиально не решает указанные проблемы.

1.2 Метод конечных элементов

Метод конечных элементов применяется в различных областях математического моделирования. Его можно использовать и для симуляции ткани [4]. Полотно разбивают на конечные элементы (обычно треугольной формы), а затем для каждого из них составляют отдельную систему линейных уравнений, описывающую действие внутренних и внешних сил на его вершины. Наконец, локальные СЛАУ объединяют в глобальную, которую решают тем или иным способом (обычно пользуются методом сопряженных градиентов).

Преимуществом этого подхода является высокое качество симуляции. Например, есть возможность моделировать ткань конкретного типа, задав ее физические параметры. Однако, как говорилось выше, на каждом шаге требуется решать СЛАУ большой размерности. Существуют способы многопоточного решения таких систем [5], однако скорость работы программы все равно оставляет желать лучшего.

Отдельно упомянем метод Бараффа и Виткина [6], бывший прорывным для своего времени и обретший огромную популярность в промышленности и области визуальных эффектов. Так, компания Pixar применяла его для расчетов поведения одежды мультипликационных персонажей [7]. Пример такого персонажа приведен на Рисунке 3. Как доказано в [8], алгоритм Бараффа и Виткина является не чем иным как необычным изложением метода конечных элементов.

Рисунок 3 – Метод Бараффа и Виткина

2 Метод PBD

Перейдем к описанию метода PBD. Ткань представляется как множество материальных точек $\bar{x}_i, i=\overline{1,n}$, с массой $m_i, i=\overline{1,n}$ на которые действуют ограничения, описывающие внутренние силы ткани [9]. Они задаются в виде скалярных функций $C_j(\bar{X}_j), j=\overline{1,k}$, зависящих от некоторых вершин $\overline{X_j}=[\bar{x}_{j1}{}^T, \bar{x}_{j2}{}^T, ..., \bar{x}_{jn_j}{}^T]^T, n_j \le n$. Связи делятся на две категории:

- условие выполнено, если $C_j(\bar{X}_j)=0$ (двустороннее),

- условие выполнено, если $C_j(\bar{X}_j)\ge 0$ (одностороннее).

Подробное изложение свойств $C_j(\bar{X}_j)$ будет приведено ниже. Сейчас же разберем общий алгоритм. На Рисунке 4 приведен псевдокод программы.

```
(1)  for all vertices i
(2)     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
(3)  end for
(4)  loop
(5)     for all vertices i
(6)        v_i = v_i + Δt·w_i·f_ext(x_i)
(7)        p_i = x_i + Δt·v_i
(8)     end for
(9)     GenerateCollisions(x_i, p_i)
(10)    for n_iter
(11)       for all constraints j
(12)          calculate ΔP_j
(13)          P_j = P_j + ΔP_j
(14)       end for
(15)    end for
(16)    for all vertices i
(17)       v_i = (p_i − x_i) / Δt
(18)       x_i = p_i
(19)    end for
(20)    EvaluateFriction(v_i)
(21) end loop
```

Рисунок 4 – Метод PBD

В строках (1)-(3) задаются начальные условия, а именно $\bar{x}_i$ и $\bar{v}_i$ в момент времени $t=0$, а также обратная масса $w_i=\dfrac{1}{m_i}$. Затем следует главный цикл программы. На очередном шаге симуляции скорости вершин обновляются по формуле

$$\bar{v}_i = \bar{v}_i + \Delta t \cdot w_i \cdot \overline{f_{ext}}(\bar{x}_i), \quad (1)$$

где

$\overline{f_{ext}}(\bar{x}_i)$ - сумма всех внешних сил, действующих на $\bar{x}_i$.

Далее движение вершин интегрируется явным методом Эйлера

$$\bar{p}_i = \bar{x}_i + \Delta t \cdot \bar{v}_i. \quad (2)$$

Таким образом, вычисляются предполагаемые (пробные) координаты $\bar{p}_i$, куда точки переместились бы без учета реакции внутренних сил. Описанные выше действия соответствуют строкам (5)-(8).

Здесь и далее используется обозначение $C_j(\bar{P}_j)$, где $\bar{P}_j = [\bar{p}_{j1}{}^T, \bar{p}_{j2}{}^T, ..., \bar{p}_{jn_j}{}^T]^T, n_j \le n$ вместо $C_j(\bar{X}_j)$, так как на практике значения этих функций вычисляются только от пробных координат вершин.

В строке (9) процедура GenerateCollisions находит коллизии ткани с самой собой и другими объектами сцены, создавая затем для каждой из них ограничение $C_j(\bar{P}_j), j = \overline{k+1, k+col}$. Наконец, в строках (10)-(15) применяются внутренние $C_j(\bar{P}_j), j = \overline{1,k}$ и внешние $C_j(\bar{P}_j), j = \overline{k+1, k+col}$ ограничения. В теле процедуры EvaluateConstraints последовательно обходятся функции $C_j(\bar{P}_j)$ и вершины $\bar{P}_j$ перемещаются, так, чтобы очередное ограничение было удовлетворено. Однако, уточняя координаты $\bar{P}_a$ так, чтобы выполнялось условие $C_a(\bar{P}_a)$ может нарушиться $C_b(\bar{P}_b)$, где $P_a \cap P_b \neq \varnothing$. Из-за этого приходится совершать несколько уточняющих итераций, многократно вызывая EvaluateConstraints. Практика показывает, что с каждым повторением коррекция становится все меньше, то есть модель к состоянию равновесия.

Когда выполнено заданное количество итераций, состояние системы обновляется

$$\bar{v}_i = \frac{\bar{p}_i - \bar{x}_i}{\Delta t}, \quad (3)$$

$$x_i = p_i, \quad (4)$$

что соответствует строкам (13)-(16). Наконец, в строке (17) скорости точек уточняются с учетом трения, которое возникло при коллизиях.

Вернемся к обсуждению ограничений $C_j(\bar{P}_j)$, которые являются ключевым элементом данного подхода. Функции $C_j(\bar{P}_j)$ строятся так, чтобы выполнялись следующие свойства:

а) $C_j(\bar{P}_j)$ имеет экстремум $C_j(\bar{P}_j) = 0$,

б) значение $C_j(\bar{P}_j)$ не зависит от движения ткани как единого целого, то есть от перемещения и вращения.

Из а) следует, что, если в данный момент времени условие $C_j(\bar{P}_j)$ не выполнено, необходимо найти точку экстремума этой функции. Это и будет искомым положением $\bar{P}_j$.

Пусть в момент времени t $C_j(\bar{P}_j) \neq 0$ и все вершины $\bar{P}_j$, имеют одинаковые массы. Покажем, как нужно их переместить, чтобы восстановить равенство. Введем обозначение $\Delta\overline{P_j} = [\Delta\bar{p}_{j1}{}^T, \Delta\bar{p}_{j2}{}^T, ..., \Delta\bar{p}_{jn_j}{}^T]^T, n_j \leq n$ - коррекция, которая позволит восстановить равенство. С учетом нового обозначения

$$C_j(\bar{P}_j + \Delta\bar{P}_j) = C_j(\bar{P}_j) + \nabla_{\bar{P}} C_j(\bar{P}_j) \cdot \Delta\bar{P}_j + O(|\Delta\bar{P}_j|^2) = 0. \qquad (5)$$

Предположим, что вектор $\Delta\bar{P}_j$ параллелен градиенту $\nabla C_{\bar{P}_j}(\bar{P}_j)$, т.е.

$$\Delta\bar{P}_j = \lambda \cdot \nabla_{\bar{P}_j} C_j(\bar{P}_j), \qquad (6)$$

где

λ - пока неизвестный коэффициент.

Тогда, подставляя (6) в (5)

$$C_j(\bar{P}_j) + \nabla_{\bar{P}_j} C_j(\bar{P}_j) \cdot \lambda \cdot \nabla_{\bar{P}_j} C_j(\bar{P}_j) = 0, \qquad (7)$$

$$\lambda = -\frac{C_j(\bar{P}_j)}{|\nabla_{\bar{P}_j} C_j(\bar{P}_j)|^2}. \qquad (8)$$

Наконец, из (8) и (6), находим общую формулу для перемещений

$$\Delta\bar{P}_j = -\frac{C_j(\bar{P}_j)}{|\nabla_{\bar{P}_j} C_j(\bar{P}_j)|^2}\nabla_{\bar{P}_j} C_j(\bar{P}_j)\,. \qquad (9)$$

Применительно к отдельной вершине $\bar{p}_{ji}$

$$\Delta\bar{p}_{ji} = \lambda_j \cdot \nabla_{\bar{p}_{ji}} C_j(\bar{P}_j)\,, \qquad (10)$$

где

$\lambda_j = -\dfrac{C_j(\bar{P}_j)}{\sum\limits_{i=1}^{n_j} |\nabla_{\bar{p}_{ji}} C_j(\bar{P}_j)|^2}$ - коэффициент, общий для всех элементов $\bar{P}_j$.

Пусть теперь точки $\bar{P}_j$ имеют разные массы $m_{ji}, i = \overline{1, n_j}, n_j \le n$. Тогда уравнение (6) примет вид

$$\Delta\bar{P}_j = \lambda_j \cdot \bar{W}_j \cdot \nabla_{\bar{P}_j} C_j(\bar{P}_j)\,, \qquad (11)$$

где

$\bar{W}_j = [w_{j1}, w_{j1}, w_{j1}, w_{j2}, w_{j2}, w_{j2}, \ldots w_{jn_j}, w_{jn_j}, w_{jn_j}]^T$ - вектор обратных масс.

Проведя преобразования, аналогичные предыдущему случаю, находим

$$\Delta\bar{p}_{ji} = \lambda_j \cdot w_{ji} \cdot \nabla_{\bar{p}_{ji}} C_j(\bar{P}_j)\,, \qquad (12)$$

где

$\lambda_j = -\dfrac{C_j(\bar{P}_j)}{\sum\limits_{i=1}^{n_j} w_{ji} \cdot |\nabla_{\bar{p}_{ji}} C_j(\bar{P}_j)|^2}$ - коэффициент, общий для всех элементов $\bar{P}_j$.

Таким образом, найдена коррекция, точек $\bar{P}_j$, которая позволяет восстановить равенство $C_j(\bar{P}_j) = 0$. Аналогичные рассуждения справедливы и для условий типа $C_j(\bar{P}_j) \ge 0$.

Важно заметить, что так как внутренние связи $C_j(\bar{P}_j)$ перемещают точки $\bar{P}_j$ вдоль градиентов $\nabla_{\bar{P}_j} C(\bar{P}_j)$, то силовое поле, создаваемое каждым их этих ограничений, будет потенциальным [10]. Таким образом, в отсутствие дополнительных сил, система вершин $\bar{x}_i, i = \overline{1,n}$, на которую действуют внутренние связи $C_j(\bar{P}_j), j = \overline{1,k}$ является замкнутой и консервативной. Следовательно, в ней выполняется закон сохранения энергии.

PBD позволяет также варьировать силу ограничений. Для функции $C_j(\bar{P}_j)$ можно ввести коэффициент жесткости $k_j \in [0;1]$, и умножать коррекцию (12) на это число. Таким образом, коррекция, которую выполняет условие, будет применяться не в полной мере. Однако, стоит помнить о том, что согласно принципу PBD уточнение координат вершин выполняется итерационно. Следовательно, более разумным было бы использовать [11]

$$k'_j = 1 - (1 - k_j)^{1/n_{iter}}, \tag{13}$$

где

n_{iter} - количество итераций.

Окончательно, (12) принимает вид

$$\Delta\bar{p}_{ji} = k'_j \cdot \lambda_j \cdot w_{ji} \cdot \nabla_{\bar{p}_{ji}} C_j(\bar{P}_j). \tag{14}$$

Оригинальный метод PBD имеет существенные недостатки. Так, результат моделирования зависит от временного шага и количества уточняющих итераций. Как и в случае представления ткани как системы масс и пружин, эти параметры влияют на жесткость материала из-за чего вновь приходится «угадывать» параметры симуляции. Метод Extended Position Based Dynamics стал развитием PBD, призванным устранить эту проблему.

3 Метод XPBD

Метод Extended Position Based Dynamics был предложен теми же авторами спустя 9 лет после создания алгоритма, описанного в прошлом разделе, и является его логическим развитием [12]. Главное преимущество XPBD над PBD заключается в устранении зависимости результатов симуляции от шага по времени и количества уточняющих итераций. Покажем, на каких физических принципах основан новый подход.

Запишем закон движения системы вершин $\bar{X} = [\bar{x}_1^T, \bar{x}_2^T, ..., \bar{x}_n^T]^T$ в потенциальном поле

$$M \cdot \ddot{\bar{X}} = -\nabla_{\bar{X}} U^T(\bar{X}), \tag{15}$$

где

M - блочная диагональная матрица масс материальных точек,

$U(\bar{X})$ - потенциальная энергия системы.

Потенциальную функцию $U(\bar{X})$ можно представить как скалярный квадрат вектора ограничений $\bar{C} = [C_1(\bar{X}), C_2(\bar{X}), ..., C_m(\bar{X})]^T$

$$U(\bar{X}) = \frac{1}{2} \cdot \bar{C}^T(\bar{X}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}), \tag{16}$$

где

α - блочная матрица, на диагонали которой расположены обратные жесткости связей.

После подставновки (16) в (15)

$$M \cdot \ddot{\bar{X}} = -\nabla_{\bar{X}} \bar{C}^T(\bar{X}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}). \tag{17}$$

Запишем разностный аналог [13] уравнения (17)

$$M \cdot \left(\frac{\bar{X}^{n-1} - 2 \cdot \bar{X}^n + \bar{X}^{n+1}}{\Delta t^2} \right) = -\nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}^{n+1}). \tag{18}$$

Обозначим

$$\bar{\lambda}^{n+1} = -\tilde{\alpha}^{-1} \cdot \bar{C}(\bar{X}^{n+1}), \tag{19}$$

где

$\tilde{\alpha} = \dfrac{\alpha}{\Delta t^2}$ - жесткость ограничения с поправкой на частоту обновления.

Тогда (18) можно записать в виде

$$M \cdot \left(\bar{X}^{n-1} - 2 \cdot \bar{X}^{n} + \bar{X}^{n+1} \right) = -\nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \bar{\lambda}^{n+1}. \tag{20}$$

Введем также вектор $\bar{P}^n = 2 \cdot \bar{X}^n - \bar{X}^{n-1} = \bar{X}^n + \Delta t \cdot \bar{V}^n$ - предполагаемое положение точек на следующем шаге. Теперь (20) представляется в форме нелинейной системы

$$M \cdot \left(\bar{X}^{n+1} - \bar{P}^n \right) - \nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \bar{\lambda}^{n+1} = \bar{0}, \tag{21}$$

$$\bar{C}(\bar{X}^{n+1}) + \tilde{\alpha} \cdot \bar{\lambda}^{n+1} = \bar{0}, \tag{22}$$

Которую решают решать методом Ньютона [14]. Обозначим уравнения (21) и (22) как $\bar{g}(\bar{X}, \bar{\lambda}) = \bar{0}$ и $\bar{h}(\bar{X}, \bar{\lambda}) = \bar{0}$ соответственно. Ниже будем использовать обозначение i вместо n, так как речь будет идти об итерационном процессе. После линеаризации получим следующие уравнения

$$\begin{bmatrix} K & -\nabla_{\bar{X}} \bar{C}^T(\bar{X}^i) \\ \nabla_{\bar{X}} \bar{C}(\bar{X}^i) & \tilde{\alpha} \end{bmatrix} \cdot \begin{bmatrix} \Delta \bar{X} \\ \Delta \bar{\lambda} \end{bmatrix} = - \begin{bmatrix} \bar{g}(\bar{X}^i, \bar{\lambda}^i) \\ \bar{h}(\bar{X}^i, \bar{\lambda}^i) \end{bmatrix}, \tag{23}$$

где

$K = \dfrac{\partial \bar{g}}{\partial \bar{X}}$ - частная производная уравнения (21) по $\bar{X}$.

На каждой итерации можно решать эту СЛАУ для $\Delta \bar{X}, \Delta \bar{\lambda}$ и обновлять координаты вершин, также коэффициенты по формулам

$$\bar{X}^{i+1} = \bar{X}^i + \Delta \bar{X}, \tag{24}$$

$$\bar{\lambda}^{i+1} = \bar{\lambda}^i + \Delta \bar{\lambda}. \tag{25}$$

Однако, вычисление матрицы системы (23), а особенно $\bar{K}$, будет занимать довольно много процессорного времени.

Примем следующие допущения:

- $K \approx M$, что влечет за собой локальную ошибку порядка $O(\Delta t^2)$ при решении задачи. Эта аппроксимация может изменить порядок сходимости метода, но не увеличивает итоговую погрешность

- $\bar{g}(\bar{X}^i, \bar{\lambda}^i) = \bar{0}$. Это предположение выполняется на первой итерации, когда $\bar{X}^0 = \bar{P}$ и $\bar{\lambda}^0 = \bar{0}$. Если же градиент вектора ограничений изменяется медленно, то величина $\bar{g}(\bar{X}^i, \bar{\lambda}^i)$ и дальше будет оставаться пренебрежимо малой.

С учетом вышеизложенного, система (23) принимает вид

$$\begin{bmatrix} M & -\nabla_{\bar{X}} \bar{C}^T(\bar{X}^i) \\ \nabla_{\bar{X}} \bar{C}(\bar{X}^i) & \tilde{\alpha} \end{bmatrix} \cdot \begin{bmatrix} \Delta\bar{X} \\ \Delta\bar{\lambda} \end{bmatrix} = -\begin{bmatrix} \bar{0} \\ \bar{h}(\bar{X}^i, \bar{\lambda}^i) \end{bmatrix}. \tag{26}$$

Используя дополнение Шура [15], можно получить выражение для $\Delta\bar{\lambda}$

$$[\nabla \bar{C}_{\bar{X}}(\bar{X}^i) \cdot M^{-1} \cdot \nabla \bar{C}_{\bar{X}}(\bar{X}^i)^T + \tilde{\alpha}] \cdot \Delta\bar{\lambda} = -\bar{C}(\bar{X}^i) - \tilde{\alpha} \cdot \bar{\lambda}^i, \tag{27}$$

откуда

$$\Delta\bar{\lambda} = \frac{-\bar{C}(\bar{X}^i) - \tilde{\alpha} \cdot \bar{\lambda}^i}{\nabla \bar{C}_{\bar{X}}(\bar{X}^i) \cdot M^{-1} \cdot \nabla \bar{C}_{\bar{X}}(\bar{X}^i)^T + \tilde{\alpha}}. \tag{28}$$

Тогда коррекция $\Delta\bar{X}$

$$\Delta\bar{X} = M^{-1} \cdot \nabla_{\bar{X}}(\bar{X}^i)^T \cdot \Delta\bar{\lambda}\,. \tag{29}$$

И хотя (28), (29) нельзя назвать точным решением задачи, на практике погрешность очень мала.

Вернемся к XPBD. Запишем (28), (29) применительно к отдельной функции $C_j(\bar{X}_j)$

$$\Delta\lambda_j = \frac{-C_j(\bar{X}_j^n) - \tilde{\alpha}_j \cdot \lambda_j}{\nabla C_{\bar{X}_j}(\bar{X}_j^n) \cdot M_j^{-1} \cdot \nabla C_{\bar{X}_j}(\bar{X}_j^n)^T + \tilde{\alpha}_j}, \tag{30}$$

$$\Delta \bar{x}_{ji} = w_{ji} \cdot \nabla_{\bar{x}_{ji}} (\bar{X}_j^n)^T \cdot \Delta\lambda_j, \qquad (31)$$

где

$\tilde{\alpha}_j = \frac{\alpha_j}{\Delta t^2} = \frac{1}{\Delta t^2 \cdot k_j}$ - поправка на шаг по времени,

M_j^{-1} - блочная матрица обратных масс вершин, на которые действует связь $C_j(\bar{X}_j)$.

Таким образом, коррекции (30) и (31) являются аналогом уравнения (12) для PBD. Наконец, приведем общий алгоритм метода XPBD.

```
(1)  for all vertices i
(2)     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
(3)  end for
(4)  loop
(5)     for all vertices i
(6)        v_i = v_i + Δt·w_i·f_ext(x_i)
(7)        p_i = x_i + Δt·v_i
(8)     end for
(9)     GenerateCollisions(x_i, p_i)
(11)    for all constraints i
(12)       λ_i = 0
(13)    end for
(10)    for n_iter
(11)       for all constraints j
(12)          calculate Δλ_i
(12)          calculate ΔP_j
(14)          λ_j = λ_j + Δλ_j
(13)          P_j = P_j + ΔP_j
(14)       end for
(15)    end for
(16)    for all vertices i
(17)       v_i = (p_i − x_i) / Δt
(18)       x_i = p_i
(19)    end for
(20)    EvaluateFriction(v_i)
(21) end loop
```

Рисунок 5 – Метод XPBD

Подведем итог. Далее вновь будем использовать обозначение $C_j(\bar{P}_j), j = \overline{1,k}$ вместо $C_j(\bar{X}_j), j = \overline{1,k}$. Метод XPBD в целом похож на PBD, но для каждого внутреннего

ограничения $C_j(\bar{P}_j), j=\overline{1,k}$ вводится обратная жесткость $\alpha_j = \frac{1}{k_j}$, на основе которой вычисляется компенсирующий частоту обновления коэффициент $\tilde{\alpha}_j = \frac{\alpha_j}{\Delta t^2}$. В начале очередного шага симуляции λ_j принимается равным нулю, а затем на каждой уточняющей итерации вычисляется поправка

$$\Delta\lambda_j = \frac{-C_j(\bar{P}_j) - \tilde{\alpha}_j \cdot \lambda_j}{\sum_{i=1}^{n_j} w_{ji} \cdot |\nabla_{\bar{p}_{ji}} C_j(\bar{P}_j)|^2 + \tilde{\alpha}_j}. \quad (32)$$

и выполняется коррекция предполагаемых координат вершин

$$\Delta\bar{p}_{ji} = \Delta\lambda_j \cdot w_{ji} \cdot \nabla_{\bar{p}_{ji}} C_j(\bar{P}_j), \quad (33)$$

после чего значение λ обновляется по формуле

$$\lambda_j = \lambda_j + \Delta\lambda_j. \quad (34)$$

Дополнительное слагаемое $\tilde{\alpha}_j$ в знаменателе (32) играет роль ограничителя силы, которую $C_j(\bar{P}_j)$ прикладывает к системе точек. Итоговое значение λ_j характеризует суммарное количество силы, приложенное связью к своим вершинам после n_{iter} итераций. Таким образом, удается преодолеть недостатки оригинального метода. Заметим, что при $\alpha_j = 0$, то есть в случае бесконечной жесткости, метод XPBD вырождается в оригинальный PBD.

В следующем разделе рассмотрены основные типы ограничений, применимых для симуляции ткани.

4 Ограничения, применяемые в XPBD для симуляции ткани

Ограничения, применяемые при симуляции ткани методом XPBD, делятся на четыре категории:

- внутренние,
- созданные коллизиями,
- пользовательские,
- фантомные.

При описании ограничения нужно уделить особое внимание на вид соответствующей функции $C(\bar{X})$, ее градиенту, а также дополнительным параметрам, от которых она зависит. Сводная таблица связей приведена в ПРИЛОЖЕНИИ Б.

4.1 Ограничение на растяжение

Это внутреннее ограничение, которое действует на две соседние вершины ткани, поддерживая между ними заданное расстояние, и обладая при этом некоторой жесткостью. Данной связи соответствует функция

$$C(\bar{P}_1, \bar{P}_2) = |\bar{P}_{1,2}| - d, \tag{35}$$

где

$\bar{P}_{1,2} = \bar{P}_1 - \bar{P}_2$ - вектор, идущий от $\bar{P}_2$ к $\bar{P}_1$,

d - оптимальное расстояние между точками.

Частные производные ограничения равны

$$\nabla_{\bar{P}_1} C(\bar{P}_1, \bar{P}_2) = \bar{n}, \tag{36}$$

$$\nabla_{\bar{P}_2} C(\bar{P}_1, \bar{P}_2) = -\bar{n}, \tag{37}$$

$$\bar{n} = \frac{\bar{P}_{1,2}}{|\bar{P}_{1,2}|}. \tag{38}$$

Таким образом, в случае использования метода PBD, коррекцию координат точек можно найти по формулам

$$\Delta\bar{P}_1 = -\frac{w_1}{w_1 + w_2} \cdot (|\bar{P}_{1,2}| - d) \cdot \bar{n}, \tag{39}$$

$$\Delta\bar{P}_2 = \frac{w_2}{w_1 + w_2} \cdot (|\bar{P}_{1,2}| - d) \cdot \bar{n} \,. \tag{40}$$

На Рисунке 6 приведена иллюстрация для этой связи.

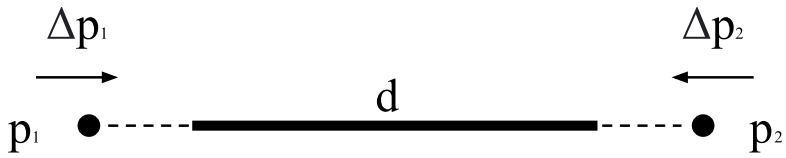


Рисунок 6 – ограничение на растяжение

4.2 Реалистичное ограничение на растяжение

Это внутреннее ограничение, основанное на механике сплошных сред [16]. Оно действует на треугольник ткани, стремясь сохранить его форму. Для корректной работы данной связи этапе инициализации симуляции требуется задать тензор модулей упругости материала C [17]

$$C = \frac{1}{1 - \nu_{уток} \cdot \nu_{основа}} \begin{pmatrix} k_{уток} & k_{уток} \cdot \nu_{основа} & 0 \\ k_{основа} \cdot \nu_{уток} & k_{основа} & 0 \\ 0 & 0 & G_{ткани} \end{pmatrix}, \tag{41}$$

где

$\nu_{уток}, \nu_{основа}$ - коэффициенты Пуассона,

$E_{уток}, E_{основа}$ - модули Юнга,

$G_{ткани}$ - модуль на сдвиг.

Также необходимо вычислить площадь $A_{нач}$ и матрицу формы каждого конечного элемента в состоянии покоя $D_{нач}$

$$D_{нач} = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix}, \tag{42}$$

где

x_i, y_i - координаты вершины на плоскости треугольника.

При применении ограничения нужно сначала найти его градиент деформации F конечного элемента

$$F = D_{текущ} \cdot D_{нач}^{-1}, \quad (43)$$

где

$D_{текущ}$ - матрица формы треугольника [18] в текущий момент времени,

а затем вычислить тензор деформации Коши-Грина ε

$$\varepsilon = \frac{1}{2} \cdot (F^T \cdot F - E). \quad (44).$$

Заметим, что ε является симметричной матрицей второго ранга, однако этот тензор можно представить в виде столбца по следующему правилу [19]

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} a \\ c \\ b \end{pmatrix}. \quad (45)$$

данный принцип действует в обе стороны. Таким образом, находят тензор напряжений S

$$S = C \cdot \varepsilon, \quad (46)$$

который является столбцом их трех элементов. К нему применятся правило, описанное выше, но в обратную сторону.

Теперь необходимо вычислить плотность энергии напряжений

$$\psi_s = \frac{1}{2} \cdot tr(\varepsilon^T \cdot S), \quad (47)$$

где

$tr(\cdot)$ - след матрицы.

Наконец, найдем энергию напряжений в треугольнике, которая, равна потенциальной энергии, накопленной в конечном элементе вследствие деформации

$$C(P_1, P_2, P_3) = A_{нач} \cdot \psi_s. \quad (48)$$

Частные производные этой функции равны

$$[\nabla_{\bar{P}_1} C(\bar{P}_1, \bar{P}_2, \bar{P}_3), \nabla_{\bar{P}_2} C(\bar{P}_1, \bar{P}_2, \bar{P}_3)] = A_{нач} \cdot P \cdot D_{тек}^{\ -T}, \quad (49)$$

$$\nabla_{\bar{P}_3} C(\bar{P}_1, \bar{P}_2, \bar{P}_3) = -\sum_{i=1}^{2} \nabla_{\bar{P}_i} C(\bar{P}_1, \bar{P}_2, \bar{P}_3), \quad (50)$$

где

$P = F \cdot S$ - тензор напряжений Пиолы-Кирхгоффа.

Полученные производные являются векторами на поверхности треугольника. Требуется перевести их в трехмерное пространство сцены, а затем скорректировать координаты вершин $\bar{P}_1, \bar{P}_2, \bar{P}_3$.

4.3 Ограничение на изгиб

Это внутреннее ограничение, которое действует на два треугольника, имеющих общее ребро, и стремится переместить их вершины так, чтобы они лежали в одной плоскости. Таким образом, соответствующая функция $C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4)$ зависит от четырех точек, где $[\bar{P}_1, \bar{P}_3, \bar{P}_2]$ принадлежат первому конечному элементу, а $[\bar{P}_1, \bar{P}_2, \bar{P}_4]$ - второму.

$$C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = \arccos\left(\frac{\bar{P'}_2 \times \bar{P'}_3}{|\bar{P'}_2 \times \bar{P'}_3|} \bullet \frac{\bar{P'}_2 \times \bar{P'}_4}{|\bar{P'}_2 \times \bar{P'}_4|} \right) - \phi_0, \quad (51)$$

где

$\bar{P'}_i = \bar{P}_i - \bar{P}_1$ - координаты вершины $\bar{P}_i$ в локальном пространстве,

ϕ_0 - начальный угол.

Заметим, что по построению нормали треугольников направлены в противоположные стороны. Отметим также, что функция (51) достигает экстремума, когда вершины принадлежат одной плоскости. Таким образом, начальный угол ϕ_0 используется лишь для индикации того, что конечные элементы переместились, но никак не для задания рекомендуемого угла. На практике имеет смысл выбирать $\phi_0 = 0$ или обновлять его значение после каждого шага симуляции. Наконец, обратим внимание на то, что связь не зависит от длин ребер, в литературе такие ограничения называют изометрическими.

Частные производные функции (51) равны

$$\nabla_{\bar{P}_2} = -\frac{\bar{P}_3' \times \bar{n}_2 + (\bar{n}_1 \times \bar{P}_3') \cdot d}{\sqrt{1-d^2} \cdot |\bar{P}_2' \times \bar{P}_3'|} - \frac{\bar{P}_4' \times \bar{n}_1 + (\bar{n}_2 \times \bar{P}_4') \cdot d}{\sqrt{1-d^2} \cdot |\bar{P}_2' \times \bar{P}_4'|}, \quad (52)$$

$$\nabla_{\bar{P}_3} = \frac{\bar{P}_2' \times \bar{n}_2 + (\bar{n}_1 \times \bar{P}_2') \cdot d}{\sqrt{1-d^2} \cdot |\bar{P}_2' \times \bar{P}_3'|}, \quad (53)$$

$$\nabla_{\bar{P}_4} = \frac{\bar{P}_2' \times \bar{n}_1 + (\bar{n}_2 \times \bar{P}_2') \cdot d}{\sqrt{1-d^2} \cdot |\bar{P}_2' \times \bar{P}_4'|}, \quad (54)$$

$$\nabla_{\bar{P}_1} = -\sum_{i=2}^{4} \nabla_{\bar{P}_i}, \quad (55)$$

где

$\bar{n}_1, \bar{n}_2$ - нормали треугольников,

d - скалярное произведене нормалей треугольников.

Важно помнить, что если значение $C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4)$ уже равно нулю, то при вычислении производных произойдет деление на ноль.

На Рисунке 7 приведена иллюстрация для этого ограничения.

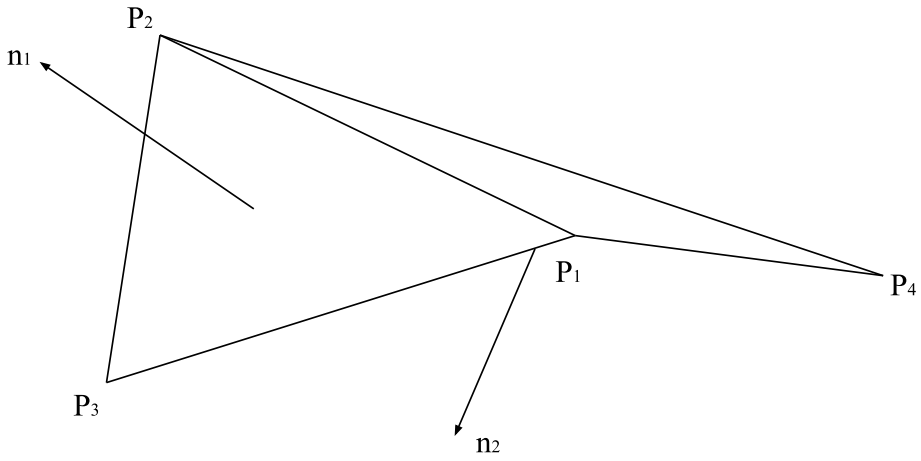


Рисунок 7 – Ограничение на изгиб

4.4 Реалистичное ограничение на изгиб

Это внутреннее ограничение, основанное на методе конечных элементов. В отличие от предыдущего, оно базируется на физических принципах, что делает симуляцию более реалистичной. Так, например, на ткани появляются достоверные складки. Данная связь тоже действует на два смежных треугольника ткани и является изометрическое.

Пронумеруем вершины и ребра треугольников, как показано на Рисунке 8.

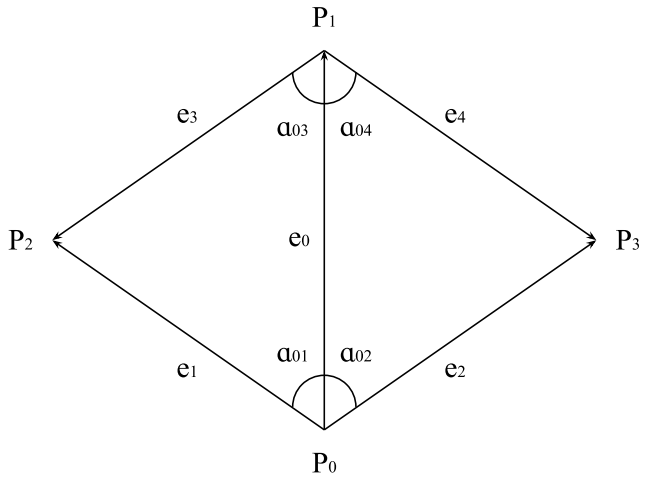


Рисунок 8 – Реалистичное ограничение на изгиб

На этапе инициализации для каждой такой пары необходимо вычислить матрицу Гесса

$$Q = \frac{3}{A_1 + A_2} \cdot \bar{K}^T \cdot \bar{K}\,, \tag{56}$$

где

A_1, A_2 - площади треугольников,

$\bar{K} = [c_{03} + c_{04}, c_{01} + c_{02}, -c_{01} - c_{03}, -c_{02} - c_{04}]$ - вектор, составленный из $c_{ij} = ctg(\alpha_{ij})$.

Тогда функция, соответствующая этому ограничению, имеет вид

$$C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = \frac{1}{2} \cdot \sum_{i,j=0}^{4} Q_{i,j} \cdot \bar{P}_i \bullet \bar{P}_j\,. \tag{57}$$

Частные производные (57) равны

$$\nabla_{\bar{P}_i} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = \sum_{j=1}^{4} Q_{i,j} \cdot \bar{P}_j\,. \tag{58}$$

4.5 Коллизия ткани с самой собой типа вершина-треугольник

Это ограничение, созданное коллизией. Как оно возникает, будет описано ниже. Сейчас же будет исходить из того, что связь уже существует. Функция C в этом случае действует на четыре вершины ткани, три из которых принадлежат одному треугольнику [20].

$$C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = \bar{n} \cdot (\bar{P}_0 - \bar{P}_1) - 2 \cdot r, \tag{59}$$

где

$\bar{P}_0$ - точка, которая может пройти сквозь треугольник,

$\bar{P}_1, \bar{P}_2, \bar{P}_3$ - вершины треугольника,

$\bar{n}$ - единичная нормаль к $[\bar{P}_1, \bar{P}_2, \bar{P}_3]$, взятая со знаком плюс, если $\bar{P}_0$ движется к нему сверху, и со знаком минус в противном случае,

r - толщина материала.

Можно видеть, что для работы ограничения необходима информация, находилась ли точка $\bar{P}_0$ на момент прошлого кадра над конечным элементом или под ним. На Рисунке 9 приведена поясняющая иллюстрация.

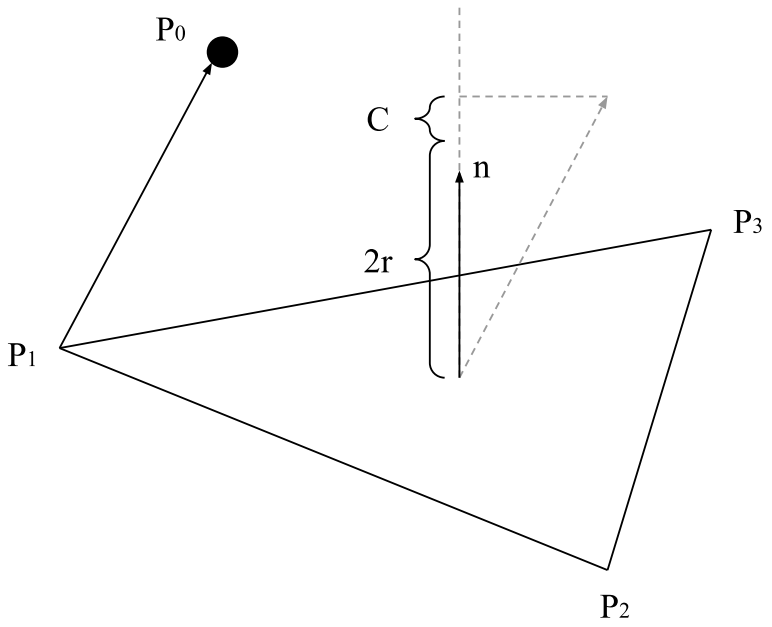


Рисунок 9 – Коллизия ткани с самой собой типа вершина-треугольник

Производные функции (59) равны

$$\nabla_{\bar{P}_0} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = \bar{n}\,, \tag{60}$$

$$\nabla_{\bar{P}_1} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_2 - \bar{P}_3) \times N \cdot \bar{n} - \bar{n}\,, \tag{61}$$

$$\nabla_{\bar{P}_2} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_3 - \bar{P}_1) \times N \cdot \bar{n} - \bar{n}\,, \tag{62}$$

$$\nabla_{\bar{P}_3} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_2 - \bar{P}_1) \times N \cdot \bar{n} - \bar{n}\,, \tag{63}$$

где

$N = \dfrac{(I - \bar{n}^T \cdot \bar{n})}{d}$ - матрица геометрической жесткости,

$d = \left|(\bar{P}_2 - \bar{P}_1) \times (\bar{P}_3 - \bar{P}_1)\right|$.

Эта связь должна иметь бесконечную жесткость, и, следовательно, вычисляется методом PBD, а не XPBD. Заметим также, что если в симуляции для каждой вершины хранится номер слоя ткани так, что наружный имеет наибольший номер, то логика ограничения упрощается. Ведь если $\bar{P}_0$ и $[\bar{P}_1, \bar{P}_2, \bar{P}_3]$ принадлежат разным слоям, то можно быстро установить, где должна находиться точка $\bar{P}_0$, над треугольником или под ним.

4.6 Коллизия ткани с самой собой типа ребро-ребро

Как и в прошлом пункте, исходим из того, что ограничение уже создано. Пусть обнаружена коллизия между ребрами $[\bar{P}_1, \bar{P}_2]$ и, $[\bar{P}_3, \bar{P}_4]$. Найдем ближайшие точки этих отрезков. На первом это будет точка $\bar{P}_\alpha$ с параметром α, а на втором – $\bar{P}_\beta$ с параметром β. Тогда связь задается функцией

$$C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = (\bar{P}_\alpha - \bar{P}_\beta) \bullet \bar{n} - 2 \cdot r\,, \tag{64}$$

где

$\bar{n}$ - единичный вектор между $\bar{P}_\alpha$ и $\bar{P}_\beta$ на предыдущем кадре,

r - толщина материала.

Производные функции (64) равны

$$\nabla_{\bar{P}_1} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -(1 - \alpha) \cdot \bar{n}\,, \tag{65}$$

$$\nabla_{\bar{P}_2} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -\alpha \cdot \bar{n}\,. \tag{66}$$

$$\nabla_{\bar{P}_3} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -(1 - \beta) \cdot \bar{n}\,, \tag{67}$$

$$\nabla_{\bar{P}_4} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -\beta \cdot \bar{n} \ . \qquad (68)$$

Это ограничение тоже должно иметь бесконечную жесткость, и вычисляться методом PBD, а не XPBD. Иллюстрация приведена на Рисунке 10.

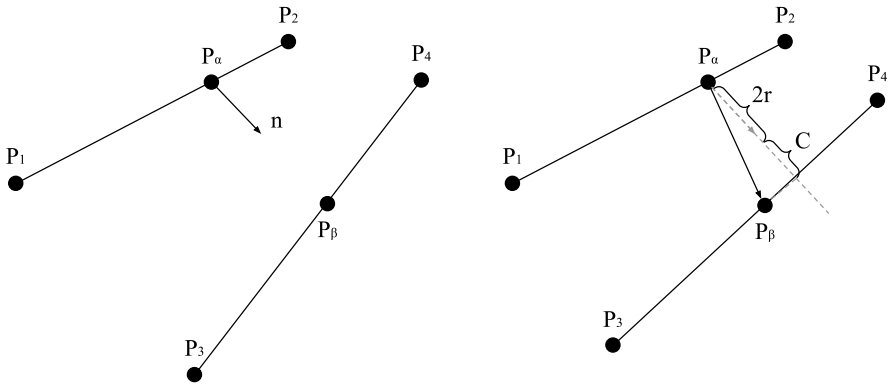


Рисунок 10 – Коллизия ткани с самой собой типа ребро-ребро

4.7 Коллизии с твердым телом типа вершина-треугольник и ребро-ребро

Вновь предполагаем, что ограничение уже создано. В таком случае, оно обрабатывается аналогично коллизии ткани с самой собой. Единственное отличие заключается в том, что масса вершин, принадлежащих твердому телу, считается бесконечной. Тогда, в формуле (14) их обратная масса, а, следовательно, и коррекция, будут равны нулю.

4.8 Фиксированный угол

Эта связь может быть создана пользователем и основана на ограничении на изгиб. Рассмотрим пример. Пусть в начальный момент времени два треугольника лежат в плоскости, то есть угол между их нормалями равен $\alpha = 180^\circ$. Требуется согнуть ткань так, чтобы его значение составило $\phi = 60^\circ$. Принимая во внимание тот факт, что функция C ограничения на изгиб стремится привести материал в плоское состояние, «виртуально» переместим вершины конечных элементов, не принадлежащие общему ребру (будем называть их свободными) так, чтобы угол стал равен $\beta = \alpha + \phi_0$. Тогда нужно повернуть свободные точки на $\gamma = \frac{\phi_0}{2} = 30^\circ$ вокруг общего ребра треугольников. Затем передаем «виртуальные» координаты вершин в функцию C ограничения на изгиб и получаем коррекции для точек. Наконец, применим эти смешения не к «виртуальным», а к реальным координатам.

Иллюстрация метода приведена на Рисунке 11

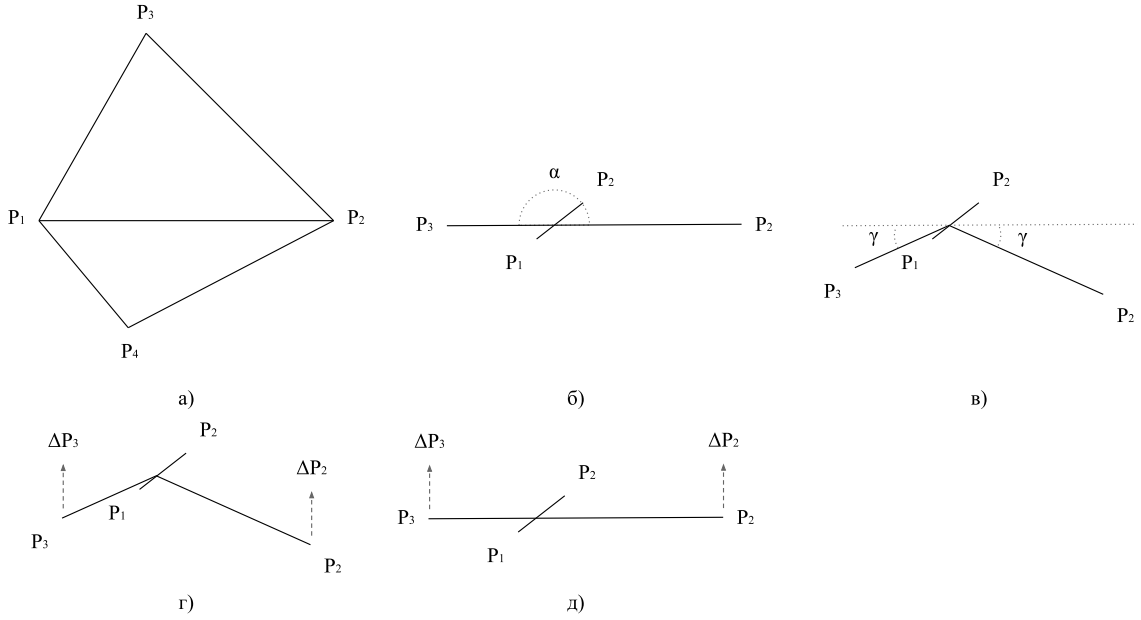


Рисунок 11 – Фиксированный угол

Выше рассмотрен случай, когда в начальный момент времени угол между конечными элементами равен $\alpha = 180^{\circ}$. Если же это условие не выполнено, перед применением метода придется выполнить еще один доворот свободных вершин, чтобы перевести треугольники в одну плоскость, а в конце алгоритма совершить обратное преобразование.

4.9 Другие ограничения, которые может задать пользователь

Кроме фиксированного угла между произвольными треугольниками ткани пользователь может задать следующие ограничения:

а) фиксированное положение вершины,

б) сшивание двух вершин.

Связь а) не имеет функции C, и коррекция, которую оно выполняет, просто перемещает точку в заданные координаты.

Работа второго ограничения состоит из двух этапов. На первом между двумя выбранными точками действует ограничение на растяжение, перемещающее их так, чтобы расстояние между ними стало равным нулю. Но, как только дистанция становится достаточно мала, происходит объединение вершин. Это и есть второй этап. Слияние происходит следующим образом: точка a выбирается ведущей, а b - ведомой. После этого любое упоминание b в ограничениях заменяется на a. Таким образом, с точки зрения

связей a и b являются одной и той же точкой. В конце каждого шага симуляции b копирует координаты a.

4.10 Фантомное ограничение

Это ограничение относится к отдельной категории. То, как оно появляется и для чего требуется, будет описано в соответствующем разделе. Здесь же укажем, что данная связь действует на переменное количество вершин $\bar{P}_1, \bar{P}_2, ..., \bar{P}_n$ (на практике до 20), находит их геометрическое среднее

$$\bar{P}_{cp} = \frac{1}{n} \cdot \sum_{i=1}^{n} \bar{P}_i \tag{69}$$

и перемещает $\bar{P}_1, \bar{P}_2, ..., \bar{P}_n$ в эту точку.

5 Разделение задачи между несколькими ядрами

5.1 Граф ограничений

Как говорилось выше, на каждом шаге симуляции необходимо обойти ограничения $C_j(\bar{P}_j), j=\overline{1,k+col}$, действующие на ткань, и скорректировать пробные координаты вершин $\bar{P}_j=[\bar{p}_{j1}{}^T,\bar{p}_{j2}{}^T,...,\bar{p}_{jn_j}{}^T]^T, n_j \le n$. Если проводить вычисления в однопоточном режиме, сложностей не возникает, однако при параллельном исполнении могут появиться ошибки. Если $C_a(\bar{P}_a)$ и $C_b(\bar{P}_b)$ влияют одну и ту же точку $\bar{P}_a \cap \bar{P}_b = \bar{P}^*$ и применяются одновременно, может возникнуть конфликт при записи нового положения $\bar{P}^*$. Такая коллизия называется гонкой потоков [21]. При этом данные, лежащие в этой ячейки памяти, будут повреждены.

Следовательно, необходимо распределить $C_j(\bar{P}_j)$ между потоками таким образом, чтобы связи из любой пары потоков не имели общих вершин. Этого можно достичь, воспользовавшись теорией графов [22]. Представим $C_j(\bar{P}_j)$ в виде вершин графа $A_j, j=\overline{1,k+col}$ и будем строить между A_a и A_b ребро B_{ab}, если $\bar{P}_a \cap \bar{P}_b \neq \varnothing$. Тогда наша проблема сводится к задаче о раскраске $G=[A,B]$ минимальным количеством цветов так, чтобы любые два смежных узла были разного цвета.

Пример графа для ткани, состоящей из шести точек, на которую действуют ограничения на растяжение, приведен на Рисунке 12.

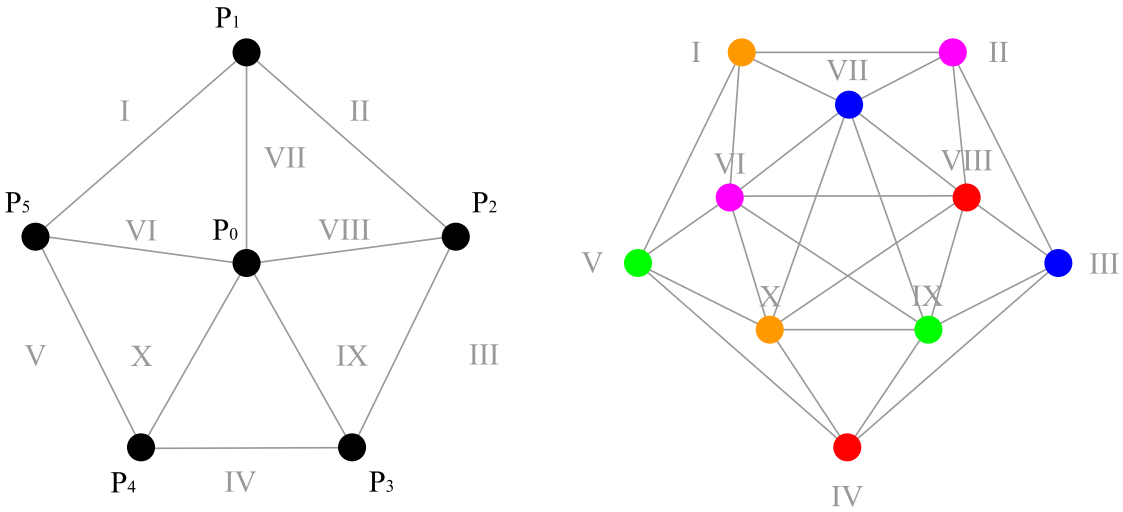


Рисунок 12 – Граф ограничений

Для раскраски G воспользуемся жадным алгоритмом [23], который принимает массив узлов A, отсортированный по вырожденности [24]. Теперь можно гарантировать,

что вершины, одного цвета могут быть распределены между потоками и применяться параллельно. Таким образом, можно разделить ограничения $C_j(\bar{P}_j)$ на «порции» $C_j = [C_{j1}(\bar{P}_{j1}), C_{j2}(\bar{P}_{j2}), ..., C_{jn_j}(\bar{P}_{jn_j})], j = \overline{1, n_{цвет}}, n_j \le k + col$ и шагать по порциям, применяя связи, принадлежащие очередной из них, параллельно.

Приведем практический пример, пусть ткань состоит из $n = 900$ вершин и подчиняется ограничениям на растяжение и изгиб. Методом, описанным выше, граф будет раскрашен в $n_{цвет} = 18$ цветов.

Однако, хотелось бы, чтобы количество «порций» было минимально возможным, ведь в этом случае наибольшее число задач можно будет выполнять параллельно. Нужно найти способ сократить количество цветов, необходимых для раскраски графа. Решением данной проблемы являются фантомные вершины.

5.2 Фантомные вершины

Вернемся к примеру на Рисунке 12. Заменим точку $\bar{P}_0$ пятью «виртуальными» вершинами $\bar{P}_0^i, i = \overline{1,5}$ и заместим в каждой из пяти функций, действовавших на $\bar{P}_0$, упоминание оригинала на соответствующую фантомную точку $\bar{P}_0^i$. Тогда из G удаляется часть ребер, число $n_{цвет}$ уменьшится. Таким образом, можно применить больше связей параллельно. Иллюстрация, поясняющая этот подход, приведена на Рисунке 13.

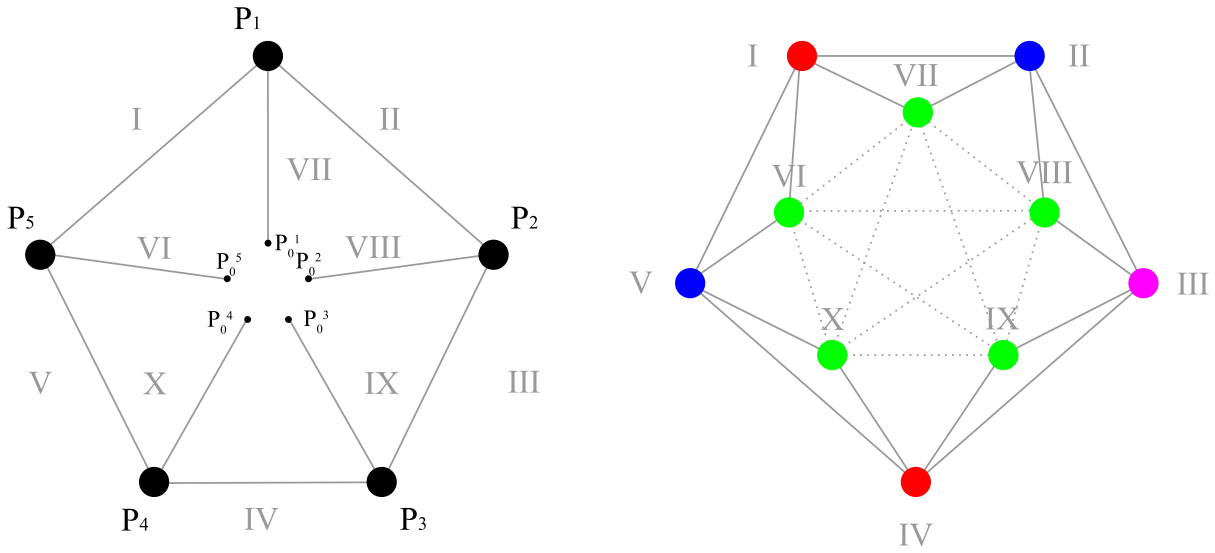


Рисунок 13 – Фантомные вершины

Доказывается, что если нужно раскрасить граф ограничений в $n_{цвет}$ цветов, достаточно найти в нем все клики $K_j = [A_{j1}, A_{j1}, ...A_{jn_j}], j = \overline{1, n}_{клик}, n_j \le k + col$, такие, что $n_j \ge n_{цвет}$, а затем для каждой клики K_j заменить общую точку ткани $P_j^* = \bigcap_{k=1}^{n_j} (\bar{P}_{jk} \mid C_{jk}(\bar{P}_{jk}) \equiv A_{jk} \in K_j), j = 1, n_{клик}, n_j \le k + col$ на фантомную. Поиск клик выполняется с помощью алгоритма Брона-Кербоша [25,26].

При использовании фантомных вершин важно после каждой итерации усреднять координаты $\bar{P}_j^i$, чтобы найти истинное положение оригинала $\bar{P}_j$. Для этого и создается специальное ограничение, описанное выше. Использование фантомных вершин вносит погрешность в симуляцию, однако ее можно свести к минимуму, уменьшив временной шаг или увеличив количество итераций.

6 Система поиска коллизий

6.1 Поиск пар-кандидатов

Напомним, что при работе с коллизиями рассматриваются следующие примитивы, из которых состоит ткань и твердые тела, с которыми она взаимодействует: вершина, ребро, треугольник. Работа системы поиска коллизий начинается после того, как движение точек проинтегрировано явным методом Эйлера и вычислены их предполагаемые координаты на новом шаге симуляции.

На первом этапе составляется массив пар-кандидатов, то есть пар примитивов, которые либо уже находятся слишком близко друг к другу, либо могут сблизится на следующем шаге. Для пространственного поиска используется R-дерево [27]. Суть этого подхода проще всего объяснить на примере. Пусть в R^2 имеются примитивы $R_i, i = \overline{8,19}$. Вокруг каждого объекта строят прямоугольник минимального размера, стороны которого идут вдоль осей координат. В зарубежной литературе его называют Axis-Aligned Bounding Box (AABB). Затем, с помощью специального алгоритма вставки формируют из AABB дерево. С помощью этой структуры данных можно быстро проверить, какие из прямоугольников пересекаются, и получить те самые пары-кандидаты. Такое R-дерево приведено на Рисунке 14.

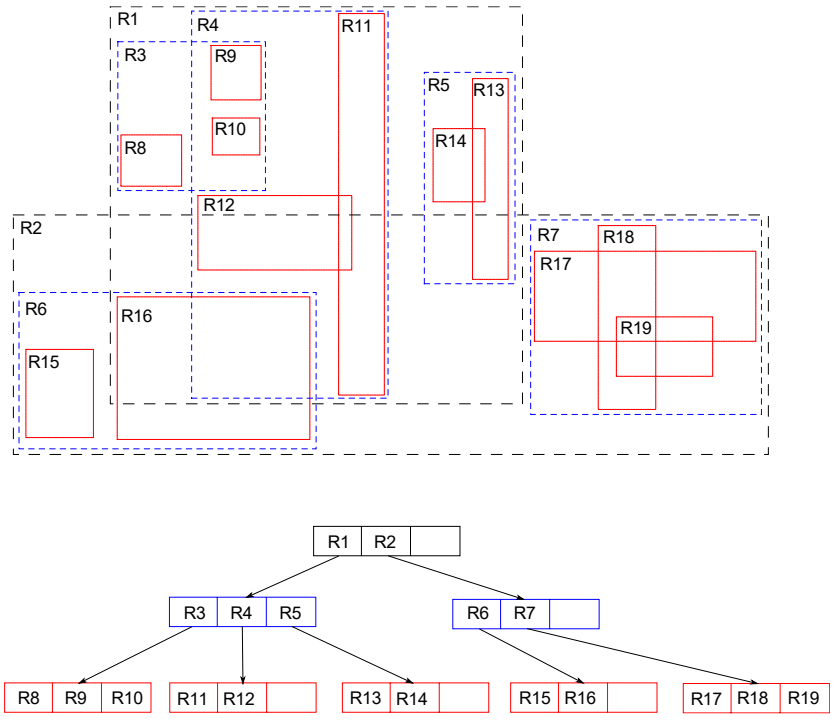


Рисунок 14 – R-дерево

На практике имеет смысл хранить в R-дереве лишь AABB треугольников ткани, и для каждого из них строить Bounding Box, который бы захватывал и текущее, и

предсказанное его положение, причем с некоторым отступом. Очевидно, что это R-дерево придется перестраивать на каждом шаге симуляции. Треугольники, из которых стоят твердые тела, можно хранить в том же R-дереве или же в отдельном.

Перейдем ко второму этапу. Каждая, найденная на первом этапе, пара треугольников порождает шесть пар-кандидатов типа вершина-треугольник и девять пар-кандидатов типа ребро-ребро. Необходимо выделить из этого множества те, между которыми действительно произойдет коллизия. Только для них создаются ограничения, описанные в одном из предыдущих разделов. Покажем, как выполняется эта проверка.

6.2 Проверка пары-кандидата типа вершина-треугольник

Пусть известно, что вершина $\bar{X}_0$ находится слишком близко к треугольнику $[\bar{X}_1, \bar{X}_2, \bar{X}_3]$, и вычислены их предполагаемые положения на следующем шаге симуляции $\bar{P}_0,[\bar{P}_1, \bar{P}_2, \bar{P}_3]$. Сначала ищут ближайшую к $\bar{X}_0$ точку на треугольнике и её естественные координаты $[a,b,c]$, ее обозначают как $\bar{X}^*$. Затем, предсказывают координаты $\bar{X}^*$ с помощью линейной интерполяции

$$\bar{P}^* = a \cdot \bar{P}_1 + b \cdot \bar{P}_2 + c \cdot \bar{P}_3 . \qquad (70)$$

Наконец, строят единичный вектор

$$\bar{n} = \frac{\bar{X}^* - \bar{X}_0}{|\bar{X}^* - \bar{X}_0|} \qquad (71)$$

и проецируют на него перемещения точек $\bar{X}_0$ и $\bar{X}^*$. Вычисляют кратчайшее расстояние d между проекциями и выполняют проверку

$$d < 2 \cdot r + r_{крит} , \qquad (72)$$

где

r - толщина ткани,

$r_{крит}$ - критическое расстояние для создания ограничения,

на основе и которой принимается решения о создании ограничения для коллизии. На Рисунке 15 приведена иллюстрация этой проверки.

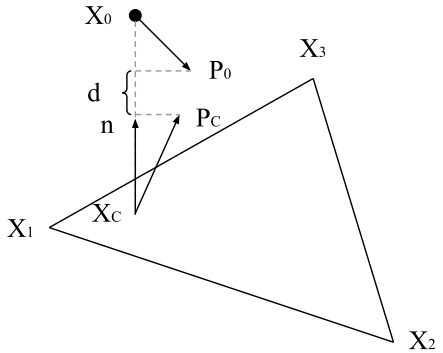


Рисунок 15 – Проверка пары-кандидата типа вершина-треугольник

6.3 Проверка пары-кандидата типа ребро-ребро

Пусть известно, что ребра $[\bar{X}_1, \bar{X}_2]$ и $[\bar{X}_3, \bar{X}_4]$ находятся слишком близко, и вычислены их предполагаемые положения на следующем шаге симуляции $[\bar{P}_1, \bar{P}_2], [\bar{P}_3, \bar{P}_4]$. Сначала находят ближайшие точки на этих отрезках. Предположим, что на $[\bar{X}_1, \bar{X}_2]$это будет $\bar{X}_\alpha$ с параметром α, а $[\bar{X}_3, \bar{X}_4]$- $\bar{X}_\beta$ с параметром β. Далее предсказывают координаты этих точек с помощью линейной интерполяции

$$\bar{P}_\alpha = (1-\alpha)\cdot \bar{P}_1 + \alpha \cdot \bar{P}_2, \tag{73}$$

$$\bar{P}_\beta = (1-\beta)\cdot \bar{P}_3 + \beta \cdot \bar{P}_4. \tag{74}$$

Наконец, строят единичный вектор

$$\bar{n} = \frac{\bar{P}_\alpha - \bar{P}_\beta}{|\bar{P}_\alpha - \bar{P}_\beta|} \tag{75}$$

и проецируют на него перемещения вершин $\bar{X}_\alpha, \bar{X}_\beta$. Ищут кратчайшее расстояние d между проекциями. Как и прошлый раз решение о создании ограничения принимается на основе условия (72). На Рисунке 16 приведена иллюстрация этой проверки.

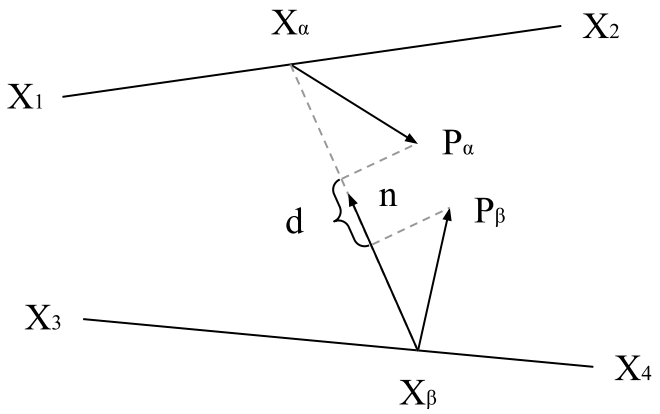


Рисунок 16 – Проверка пары-кандидата типа ребро-ребро

6.4 Использование метода Representative triangles для проверки пары-кандидата

Пусть на втором этапе удалена большая часть пар-кандидатов. Однако, среди оставшихся могут быть дубликаты. Например, на первом этапе установлено, что треугольники $[\bar{P}_1, \bar{P}_2, \bar{P}_3]$ и $[\bar{P}_1, \bar{P}_3, \bar{P}_4]$ находятся слишком близко к $[\bar{P}_5, \bar{P}_6, \bar{P}_7]$ тогда пара-кандидат, состоящая из вершины $\bar{P}_1$ и треугольника $[\bar{P}_5, \bar{P}_6, \bar{P}_7]$ может быть создана дважды. Чтобы не допустить этого, на этапе инициализации симуляции необходимо пройти по всем конечным элементам ткани и твердых тел, указав, какими из своих примитивов (ребер и вершин) владеет каждый из треугольников так, чтобы каждое ребро и каждая вершина ткани принадлежали только одному из них. Затем, при создании пар-кандидатов на основе двух, расположенных слишком близко, треугольников, необходимо создавать пары только из тех примитивов, которые принадлежат треугольникам.

7 Трение

Для симуляции трения была выбрана модель Амонтона-Кулона [28]. После очередного шага симуляции для каждой коллизии типа вершина-треугольник необходимо корректировать скорость вершины относительно треугольника $\bar{V}_{отн}$ по формуле

$$\bar{V}'_{отн} = \bar{V}_н + \bar{V}'_m , \tag{76}$$

где

$\bar{V}_н$ - нормальная составляющая скорости вершины относительно треугольника,

$\bar{V}'_m$ -скорректированная тангенциальная составляющая скорости вершины относительно треугольника.

В свою очередь, $\bar{V}'_m$ вычисляется по формуле

$$\bar{V}'_m = \max\left(1 - \mu \cdot \frac{\Delta\bar{V}_н}{|\bar{V}_m|}, 0\right) \cdot \bar{V}_m , \tag{77}$$

где

μ - коэффициент трения

$\Delta\bar{V}_н$ - изменение нормальной составляющей скорости вершины относительно треугольника, с момента прошлого кадра прошлого шага симуляции (нормальное ускорение),

$\bar{V}_m$ - тангенциальная составляющая скорости вершины относительно треугольника.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Yang J.D. Cloth Modeling Simulation Based on Mass Spring Model [Текст] / Yang J.D., Shang S.Y. // Applied Mechanics and Materials. – 2013. – №310. – С. 676-683.
2. Carvalho L. Dynamic Cloth Simulation: A Comparative Study of Explicit and Implicit Numerical Integration [Текст] / Carvahlo L., Vidal C., Cavalcante-Neto J., Oliveira S. // 14th Symposium on Virtual and Augmented Reality (SVR). – 2012. – С. 56-65.
3. Jakobsen T. Advanced character physics [Текст] / Jakobsen T. // Game Developers Conference 2001. – 2001.
4. Etzmuss O. A fast finite element solution for cloth modelling [Текст] / Etzmuss O., Keckeisen M., Strasser W. // 11th Pacific Conference on Computer Graphics and Applications. – 2003.
5. Баландин М.Ю. Методы решения СЛАУ большой размерности [Текст] / Баландин М.Ю., Шурина Э.П. – Новосибирск: Изд-во НТГУ, 2000. – 70 с.
6. Baraff D. Large Steps in Cloth Simulation [Текст] / Baraff D., Witkin A. // SIGGRAPH ’98. – 1998. – С. 43-54.
7. Baraff D. Untangling Cloth [Текст] / Baraff D., Witkin A. // SIGGRAPH ’03. – 2003. – С. 862-870.
8. Kim. T. A Finite Element Formulation of Baraff-Witkin Cloth [Текст] / Kim. T. // SCA ’20. – 2020. – С. 1-9.
9. Matthias M. Position Based Dynamics [Текст] / Matthias M., Heidelberger B., Hennix M., Ratcliff J. // VRIPHYS: 3rd Workshop in Virtual Realitiy, Interactions, and Physical Simulation. – 2006.
10. Дронг В.И. Курс теоретической механики: учебник для вузов [Текст] / Дронг В.И., Дубинин В.В., Ильин М.М. и др. – М.: Изд-во МГТУ им Н.Э. Баумана, 2005. – 736 с.
11. Bender J. A Survey on Position Based Dynamics [Текст] / Bender J., Matthias M., Miles M. // EG’17. – 2017. – С. 1-31.
12. Miles M. XPBD Position-Based Simulation of Compliant Constrained Dynamics [Текст] / Miles M., Matthias M., Nuttapong C. // MIG’16. – 2016., - С. 49-54.
13. Власова Б.А. Приближенные методы математической физики: учебник для вузов [Текст] / Власова Б.А., Зарубин В.С., Кувыркин Г.Н. – М.: Изд-во МГТУ им Н.Э. Баумана, 2001. – 700 с.
14. Аттетков А.В. Методы оптимизации: учебник для вузов [Текст] / Аттетков А.В., Галкин С.В, Зарубин В.С. – М.: Изд-во МГТУ им Н.Э. Баумана, 2000. – 440 с.
15. Прасолов В.В. Задачи и теоремы линейной алгебры [Текст] / Прасолов В.В. – М.: Изд-во МЦНМО, 2015. – 576 с.

16. Bender J. Position-Based Simulation of Continuous Materials [Текст] / Bender J., Koschier D., Charrier P., Weber D. // Computers and Graphics. – 2014. – №44. – С. 1-10.
17. Димитриенко Ю.И. Механика сплошной среды: учеб пособие [Текст]: в 4 т. / Димитриенко Ю.И. – М.: Изд-во МГТУ им Н.Э. Баумана, 2013. – 623 с.
18. MgGinty B. Finite Element Coordinate Mapping [Электронный ресурс]. URL: https://www.continuummechanics.org/finiteelementmapping.html (дата обращения 28.04.23)
19. Faure F. Introduction to Finite Elements [Электронный ресурс]. URL: http://imagine.inrialpes.fr/people/Francois.Faure/htmlCourses/FiniteElements.html (дата обращения 28.04.23)
20. Lewin. C. Cloth Self Collision with Predictive Contacts [Текст] / Lewin. C. // GDC 2018. – 2018.
21. Уильямс Э. C++. Практика Многопоточного Программирования [Текст] / Уильямс Э. – СПБ.: Изд-во Питер, 2020. – 640 с.
22. M. Fratarcangeli Scalable Partitioning for Parallel Position Based Dynamics [Текст] / M. Fratarcangeli, F. Pellacini // Computer Graphics Forum. – 2015.
23. Matula, D. Graph colouring algorithms [Текст] / Matula, D., Marble, G., Isaacson, J. // Graph Theory and Computing. – 1972. – С. 109-122.
24. Matula D.W. Smallest-last ordering and clustering and graph coloring algorithms [Текст] / Smallest-last ordering and clustering and graph coloring algorithms // Journal of the ACM. – 1983. – №30. – С. 417-427.
25. Bron C. Finding All Cliques of an Undirected Graph [Текст] / Bron C., Kerbosch J. // Communications of the ACM. – 1973. – №16. – С. 575-577.
26. Wei Y.-W. Accelerating the Bron-Kerbosch Algorithm for Maximal Clique Enumeration Using GPUs [Текст] / Wei Y.-W., Chen W.-M.,Tsai H.-H. // IEEE Transactions on Parallel and Distributed Systems. – 2021. – № 32. – С. 2352-2366.
27. Guttman A. R-Trees: A Dynamic Index Structure for Spatial Searching [Текст] / Guttman A. // SIGMOD '84. – 1984. – С. 47-57.
28. Bridson R. Robust Treatment of Collisions, Contact and Friction for Cloth Animation [Текст] / Bridson R., Fedkiw R., Anderson J. // ACM Transactions on Graphics. – 2002. - № 21. – С. 594-603.