Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 7 № 1

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2017

Тематика номера:
математические модели в физике, математическое моделирование и визуализация, информационные технологии, теоретические вопросы численного анализа.

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, вопросы численного анализа, история науки и техники

The topic of the issue:

mathematical models in physics, mathematical modeling and visualization, information technology, theoretical questions of numerical analysis.

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, problems of numerical analysis, history of science and of technique

Заведующий редакцией: Ю.Н. Штейников

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

СОДЕРЖАНИЕ

Границы применимости метода n-окрестностей для исследования модели Изинга

Б.В. Крыжановский [1], Л.Б. Литинский [2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail's:[1] kryzhanov@mail.ru, [2] litin@mail.ru.

Аннотация. Ранее был предложен метод n-окрестностей для приближенного вычисления статистической суммы. В настоящей работе метод применяется для исследования модели Изинга на D-мерной кубической решетке. При нулевом магнитном поле для произвольной размерности D решается уравнение состояния. Вводится эффективное координационное число q, характеризующее взаимодействие спина с ближайшим окружением В зависимости от величины q реализуется один трех вариантов поведения системы. Проведен исчерпывающий анализ возможных фазовых переходов. Для больших размерностей D (D>2) наши результаты хорошо согласуются с данными компьютерного моделирования.

Ключевые слова: метод n-окрестностей, модель Изинга, критическая температура.

1.Введение

Вычисление статистической суммы для модели Изинга является одной из центральных задач статистической физики. Для целого ряда содержательных моделей эту задачу удается решить точно [1-4]. Наличие «трудных» случаев, не поддающихся точному решению, привело к разработке методов, решающих задачу в том или ином приближении; обширную библиографию на эту тему можно найти в [5, 6]. Отметим, что проблема имеет и нефизические приложения: к задачам комбинаторной оптимизации [7], теории обучения машин [8] и компьютерной обработки изображений [9, 10].

В работах [11–13] нами предложен метод n-окрестностей для приближенного вычисления статистической суммы. Суть нашего подхода (не зависящего от типа матрицы связи) состоит в следующем. Зафиксируем начальную конфигурацию (начальное состояние) $\mathbf{s}_0 \in \mathbf{R}^N$, и определим ее n-окрестность Ω_n как множество состояний, отличающихся от $\mathbf{s}_0$ противоположными значениями n координат; n меняется от 0 до N ($0 \le n \le N$). Распределение энергий состояний из n-окрестности в общем случае неизвестно, однако в [12, 13] получены точные выражения для среднего значения E_n и дисперсии σ_n^2 этого распределения; E_n и σ_n^2 выражаются через характеристики матрицы связи. Эмпирический факт состоит в том, что распределение энергий состояний из Ω_n является одногорбым, и хорошо аппроксимируется гауссовой плотностью с параметрами E_n и σ_n^2 (см. [13]). Тогда в выражении $Z_N = \sum_{n=0}^{N} \sum_{\mathbf{s}\in\Omega_n} \exp\left(-\beta E(\mathbf{s})\right)$ для статистической суммы, суммирование по состояниям из Ω_n можно заменить интегрированием по гауссовой мере, а суммирование по n – интегрированием по параметру $x = n/N$ от 0 до 1. Статистическая сумма принимает вид двойного интеграла, который вычисляется методом перевала: $Z_N \approx \int dx \int dE \exp\left[-N \cdot F(x,E)\right]$. Функция $F(x,E)$ в показателе экспоненты зависит от обратной температуры β, внешнего магнитного поля H (если имеется) и матрицы связи. В общем виде выражение для $F(x,E)$ получено в [12, 13].

В настоящей работе метод n-окрестностей применяется для исследования модели Изинга на D-мерном гиперкубе. За почти 100 лет изучения модели Изинга для нее получен целый ряд точных результатов, на которых можно тестировать новые методы вычислений. Основные полученные нами результаты, таковы.

В предположении, что взаимодействие между ближайшими соседями вдоль различных направлений кубической решетки может быть различным, получено уравнение состояния, которое обобщает уравнение Брэгга-Вильямса (см. разделы 2, 3). Решетка характеризуется эффективным координационным числом q, которое описывает взаимодействие спина с ближайшим окружением. Когда взаимодействие между

ближайшими спинами одинаково, q есть просто число ближайших соседей: $q = 2D$.

Решению уравнения состояния посвящены разделы 4, 5. Для решеток больших размерностей, $D > 3$, удается воспроизвести все известные для модели Изинга результаты: наличие фазового перехода второго рода и классические значения критических показателей. Получено аналитическое выражение для критического значения обратной температуры $b_c = \left(1 - \sqrt{1 - 2/D}\right)/2$, которое хорошо описывает результаты компьютерного моделирования для размерностей $3 \le D \le 7$ [14 - 17] (раздел 4).

Для трехмерной модели Изинга ($D = 3$) наш подход позволяет с хорошей точностью воспроизвести все общепринятые результаты за исключением того, что критические показатели у нас получаются классическими; как более точный метод ренорм-группы дает для них неклассические значения [18, 19].

Для малых размерностей ($D < 3$) наш подход не работает. Для одномерной модели Изинга у нас - как и в приближении среднего поля - получается фазовый переход при конечной температуре. Для двумерной модели ($D = 2$) наш подход неплохо воспроизводит известные количественные результаты, однако предсказывает фазовый переход первого рода, что противоречит точному решению модели Изинга (раздел 5).

В заключительном разделе обсуждаются дальнейшие перспективы метода. Сложные вычисления вынесены в Приложения.

2. Обозначения и начальные выражения

Пусть $\mathbf{T} = \left(T_{ij}\right)_1^N$ матрица связи, отвечающая D-мерной модели Изинга с взаимодействием между ближайшими соседями, $\mathbf{s}_0 = (1,1,...,1) \in \mathbf{R}^N$ - основное состояние спиновой системы, а Ω_n - n-окрестность основного состояния: $\Omega_n = \left\{\mathbf{s} : \sum_{i=1}^N s_i = N - 2n\right\}$, $n = 0,\ 1,\ldots,\ N$. Наш метод вычисления статистической суммы основан на том, что распределение энергий состояний из Ω_n аппроксимируется гауссовой плотностью. Детальное обсуждение этого вопроса можно найти в [13]. Там же приведены точные выражения для среднего (по Ω_n) значения энергии E_n и дисперсии σ_n^2.

Энергия вычисляется в расчете на один спин: если H - величина однородного магнитного поля, то:

$$E(\mathbf{s}, H) = -\frac{\sum_{i \ne j}^N T_{ij} s_i s_j}{2N} - \frac{H \sum_{i=1}^N s_i}{N} = \\ = E(\mathbf{s}) - H\left(1 - \frac{2n}{N}\right). \qquad (1)$$

Через E_0 обозначим энергию основного состояния:

$$E_0 = E(\mathbf{s}_0) = -\frac{1}{2N}\sum_{i \ne j}^N T_{ij}\ . \qquad (2)$$

Устремим $N \to \infty$ и введем $x = n/N \in [0,1]$. В [13] получены асимптотические выражения для среднего значения энергии по n-окрестности и дисперсии:

$$E_x = \lim_{N\to\infty} E_n = E_0(1-2x)^2, \\ \sigma_x^2 = \lim_{N\to\infty} \sigma_n^2 = \frac{8\sum_{ij} T_{ij}^2}{N^2} x^2(1-x)^2. \qquad (3)$$

В Приложении 1 показано, что асимптотическое выражение для статистической суммы имеет вид $Z_N \sim \int_0^1 dx \int_{E_0}^{|E_0|} \exp\left[-N \cdot F(x,E)\right] dE$, где $N >> 1$, а функция в показателе экспоненты равна:

$$F(x,E) = L(x) + \beta\left[E - H(1-2x)\right] + \frac{1}{2N}\left(\frac{E - E_x}{\sigma_x}\right)^2.$$

Здесь $L(x) = x\ln x + (1-x)\ln(1-x)$, а $\beta = 1/T$ - обратная температура. Интеграл в выражении для Z_N вычисляют методом перевала, для чего необходимо отыскать глобальный минимум функции $F(x,E)$ как функцию параметра β. Иначе говоря, требуется решить систему равнений:

$$\frac{\partial F}{\partial E} = \beta + \frac{E - E_x}{N\sigma_x^2} = 0,$$

$$\frac{\partial F}{\partial x} = \ln\frac{x}{1-x} + \frac{E - E_x}{N\sigma_x}\frac{\partial}{\partial x}\left(\frac{E - E_x}{\sigma_x}\right) + 2\beta H = 0.$$

Из первого уравнения следует, что минимум достигается на таком интервале значений x, на котором выполняется неравенство: $\beta N\sigma_x^2 + E_0 - E_x \le 0$. Разрешая первое уравнение относительно β, и подставляя результат во второе, получаем, что отыскание седловой точки сводится к минимизации функции одной переменной

$$f(x) = L(x) + \beta E_x - \frac{\beta^2 N\sigma_x^2}{2} - \beta H(1-2x),$$

при условии $\beta N\sigma_x^2 + E_0 - E_x \le 0$. Используем более удобную нормировку и введем новые обозначения:

$$b=\beta\frac{\sum_{ij}T_{ij}^2}{\sum_{ij}T_{ij}},\ q=\frac{\left(\sum_{ij}T_{ij}\right)^2}{N\sum_{ij}T_{ij}^2},h=\frac{2\sum_{ij}T_{ij}}{\sum_{ij}T_{ij}^2}H\ .(4)$$

В результате задача принимает вид: для каждого значения «нормированной» обратной температуры b необходимо отыскать глобальный минимум функции

$$f(x)=L(x)-\frac{qb}{2}\left[(1-2x)^2+8b(x(1-x))^2\right]-\\-\frac{bh}{2}(1-2x) \quad (5)$$

на интервале $(0,x_b)$, где

$$x_b=\begin{cases}\frac{1}{2}, & \text{когда } b\le 1\\ \frac{1-\sqrt{1-1/b}}{2}, & \text{когда } b\ge 1.\end{cases} \quad (6)$$

В терминах обратной температуры b свободная энергия имеет вид:

$$\mathbf{f}(b)=b^{-1}\cdot\min_{x\in(0,x_b)}f(x). \quad (7)$$

Сделаем несколько замечаний. Пока $b\le 1$, функцию $f(x)$ (5) минимизируют на интервале [0,1/2], однако как только значение b переваливает за 1, правая граница интервала x_b становится переменной. Когда b стремится к бесконечности, правая граница стремится к 0: $\lim_{b\to\infty}x_b=0$.

Далее, для D-мерной модели Изинга с взаимодействием J между ближайшими соседями выражения (4) дают: $\sum_{ij}T_{ij}=DNJ$, $b=\beta J$, $q=2D$, $\sum_{ij}T_{ij}^2=DNJ^2/2$, $h=2H/J$. Большую роль в дальнейшем играет параметр q, который в простейшем случае равен числу ближайших соседей каждого спина. Если взаимодействие вдоль различных направлений D-мерной решетки различно, параметр q не обязан принимать целочисленные значения. Например, для двумерной модели Изинга с различными константами взаимодействия по вертикали J и по горизонтали K имеем:

$$q=2\left(1+\frac{2}{K/J+J/K}\right).$$

Эта характеристика может принимать любое значение из интервала $(2,4)$. Аналогично, для 3D модели Изинга с различными константами взаимодействия ($J\ne K\ne L$), имеем:

$$q=2\frac{(J+K+L)^2}{J^2+K^2+L^2}\in(2,6).$$

В общем случае q является действительным, и представляет собой эффективное координационное число, характеризующее взаимодействие спина с ближайшим окружением.

Наконец, отложим рассмотрение уравнения с магнитным полем до отдельной публикации, положив $H=0$.

3. Уравнение состояния

1) Уравнение состояния. Для каждого значения b на интервале $[0,x_b]$ требуется отыскать минимум функции

$$f(x)=L(x)-\frac{qb}{2}\left[(1-2x)^2+8b(x(1-x))^2\right], \quad (8)$$

для чего необходимо проанализировать решения уравнения

$$\frac{\partial f}{\partial x}=\ln\frac{x}{1-x}+2qb(1-2x)\left[1-4bx(1-x)\right]=0\ .(9)$$

Очевидно, что $x_0=1/2$ всегда является решением уравнения (9). Назовем решение x_0 тривиальным.

Чтобы найти остальные решения, избавимся от тривиального решения, преобразовав уравнение (9) к виду:

$$\frac{\ln\frac{1-x}{x}}{2(1-2x)}=qb\left[1-4bx(1-x)\right]. \quad (10)$$

Уравнение (10) называется уравнением состояния. Обозначим через $l(x)$ его левую часть, а через $r(x,b)$ - правую:

$$l(x)=\frac{\ln\frac{1-x}{x}}{2(1-2x)},\quad r(x,b)=qb\left[1-4bx(1-x)\right]$$

Функции $l(x)$ и $r(x,b)$ симметричны относительно точки $x_0=1/2$; во избежание путаницы будем изучать их поведение на интервале [0, 1/2].

Функция $l(x)$ монотонно убывает от $+\infty$ на левом краю интервала до 1 на правом краю: $l(x_0)=1$. Ее первая и вторая производные на правом краю равны $l'(x_0)=0$ и $l''(x_0)=8/3$ соответственно. Квадратичная функция $r(x,b)$ монотонно убывает от значения $r(0)=qb$ на левом краю до $r(x_0)=qb(1-b)$ на правом – см. штриховую линию на нижних панелях рис.1. По мере возрастания параметра b у функции $f(x)$ появляются точки перегиба и/или локальные экстремумы – см. графики на верхних панелях рис. 1. Для изучения этих трансформаций надо анализировать взаимное расположение кривых $l(x)$ и $r(x,b)$.

Пока значение b невелико, кривые $l(x)$ и $r(x,b)$ не имеют общих точек – см.

нижнюю панель рис.1a. Это означает, что функция $f(x)$ монотонно убывает от значения $f(0) = -qb/2$ на левом краю до значения $f(x_0) = -\ln 2 - qb^2/4$ на правом краю (рис.1a), а единственным решением уравнения состояния является тривиальное решение $x_0 = 1/2$. Свободная энергия (7) имеет вид:

$$\mathbf{f}(b) = \frac{f(x_0)}{b} = -\frac{\ln 2}{b} - \frac{qb}{4}, \quad b << 1. \qquad (11)$$

С ростом параметра b кривая $r(x,b)$ как целое сдвигается вверх, приближаясь снизу к кривой $l(x)$. При некотором значении параметра b кривые $l(x)$ и $r(x,b)$ коснутся друг друга. Абсцисса касания может быть либо внутренней точкой интервала $(0, x_0)$ - см. нижнюю панель рис.1b; либо кривые могут коснуться друг друга на правом конце интервала, в точке $x_0 = 0.5$ - см. нижнюю панель рис. 1c.

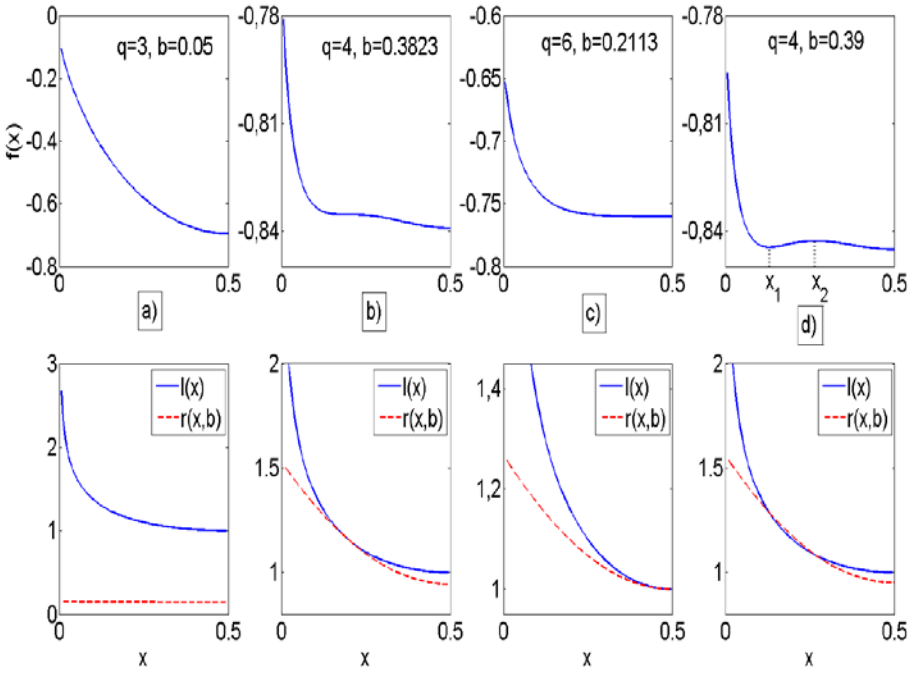


Рис. 1. Для различных значений *q* и *b* графики функций *f(x)* (верхние панели) и *l(x), r(x,b)* (нижние панели) – см. текст.

Касание кривых $l(x)$ и $r(x,b)$ во внутренней точке интервала означает, что у функции $f(x)$ появилась точка перегиба, в которой производная функции $f(x)$ равна нулю – см. верхнюю панель рис.1b. Такая точка не является минимумом функции $f(x)$, и данное решение уравнения (10) интереса не представляет. Однако стоит здесь слегка увеличить значение параметра b, и вместо точки перегиба у функции $f(x)$ появятся два экстремума: локальный минимум x_1 и локальный максимум x_2 – ср. верхние панели рис. 1b и рис. 1d. Новый локальный минимум является нетривиальным решением уравнения (9). Его глубину необходимо сравнивать с глубиной минимума в точке $x_0 = 1/2$. Выясним, при каких условиях кривые $l(x)$ и $r(x,b)$ касаются друг друга во внутренней точке интервала $(0, x_0)$.

2) Касание кривых $l(x)$ и $r(x,b)$ во внутренней точке интервала.

Такая ситуация описывается системой уравнений:

$$\begin{cases} l(x) = r(x,b) \\ l'(x) = r'(x,b) \neq 0 \end{cases}. \qquad (12)$$

Знак неравенства во втором уравнении исключает касание кривых на правом конце интервала, в точке x_0. Систему (12) удается решить, разделив переменные x и b - см. Приложение 2. Приведем окончательный результат.

Утверждение 1. Кривые $l(x)$ и $r(x,b)$ соприкасаются во внутренней точке интервала $(0, x_0)$ только для $q < 16/3$. Касание кривых

происходит в точке x_t, являющейся решением уравнения:

$$G(x) = \frac{[\xi(x) + \varphi(x)]^2}{\xi(x)} = q;$$

выражения для функций $\varphi(x)$ и $\xi(x)$ сравнительно сложны и приводятся в Приложении 2. График функции $G(x)$ показан на рис. 2 сплошной линией: $G(x)$ монотонно возрастает от 0 до значения $G(x_0) = 16/3$. Значение координаты x_t точки касания определяется пересечением графика функции $G(x)$ с прямой, параллельной оси абсцисс и отстоящей от нее на расстояние q (см. рис.2). Значение b_t параметра b, при котором происходит касание кривых $l(x)$ и $r(x,b)$, вычисляется по найденному значению x_t (индекс "t" от touch):

$$b_t = B(x_t), \text{ где } B(x) = \frac{\xi(x)}{\xi(x) + \varphi(x)}. \qquad (13)$$

График функции $B(x)$ показан на рис. 2 штриховой линией: $B(x)$ монотонно убывает от $+\infty$ на левом краю до минимального значения $B(x_0) = 0.25$.

Рис. 2 можно использовать и для решения другой задачи, например: по заданному значению $b_t > 0.25$ сначала отыскиваем координату x_t, в которой произойдет касание кривых $l(x)$ и $r(x,b)$; а уж затем определяем значение q, для которого кривые $l(x)$ и $r(x,b)$ соприкасаются в точке x_t при $b = b_t$.

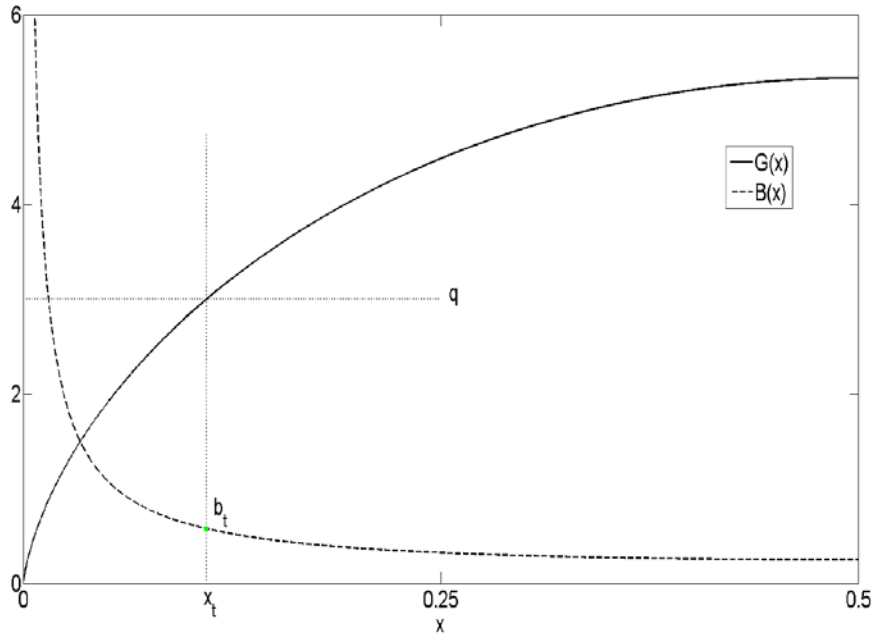


Рис.2. Графики функций *G(x), B(x)* для отыскания внутренней точки касания кривых *l(x)* и *r(x,b)* - см. текст.

4. Интервал значений $q \ge 16/3$

1. Критическая температура. Рассмотрим, что происходит, когда $q \ge 16/3$. В этом случае кривые $l(x)$ и $r(x,b)$ не имеют точек касания внутри интервала $(0, x_0)$. Остается возможность их касания на правом конце интервала, в точке $x_0 = 0.5$ (рис.1c). Условие такого касания - $l(x_0) = r(x_0, b)$ - дает квадратное уравнение $1 = qb(1-b)$, корни которого равны:

$$b_{\pm} = \frac{1 \pm \sqrt{1 - 4/q}}{2}. \qquad (14)$$

Иными словами, кривые $l(x)$ и $r(x,b)$ соприкасаются в точке $x_0 = 0.5$ дважды: когда $b = b_-$, и когда $b = b_+$. Первое касание кривых происходит, когда параметр b станет равным наименьшему из двух корней (14) – обозначим его b_c:

$$b_c = b_- = \frac{1 - \sqrt{1 - 4/q}}{2}. \qquad (15)$$

Как только b превзойдет b_c, сразу же x_0 из точки минимума $f(x)$ превращается в точку максимума:

$$f''(x_0) \sim 1 - qb(1-b) < 0, \text{ когда } b_- < b < b_+. \qquad (16)$$

В тот же момент рядом с точкой x_0 - слева от нее - появится новая точка минимума

функции $f(x)$ - обозначим ее x_1: $x_1 < x_0$ - см. верхнюю панель рис. 3a. По мере дальнейшего возрастания b точка минимума x_1 неуклонно смещается в направлении к 0, а глубина минимума будет увеличиваться. Кривая $r(x,b)$ как целое будет по-прежнему подниматься вверх. Когда значение b станет равным 0.5, правый конец кривой достигнет максимально возможной высоты: $r(x_0,1/2) = \max_b r(x_0,b) = q/4$ (рис.3b). После этого правый конец кривой $r(x_0,b)$ станет опускаться вниз, а ее левый конец $r(0,b)$ будет по-прежнему подниматься вверх. Когда b достигнет значения, равного второму корню уравнения (14) - $b = b_+$ - кривые $l(x)$ и $r(x,b)$ опять соприкоснутся в точке $x_0 = 0.5$. Едва лишь значение b превзойдет b_+, точка x_0 опять превратится в точку минимума функции $f(x)$ - см. (16). Между x_1 и x_0, в точке x_2, у функции $f(x)$ появится максимум - см. рис. 3c. Теперь функция $f(x)$ имеет два минимума – глобальный минимум в точке x_1 и локальный минимум в точке x_0. По мере дальнейшего возрастания b - до значения $b = 1$ - точка глобального минимума x_1 будет неуклонно смещаться в направлении к 0, а глубина минимума будет расти. Можно оценить разность глобального и локального минимумов для значения параметра $b = 1$: $f(x_1) - f(x_0)\big|_{b=1} < \ln 2 - q/4 < 0$.

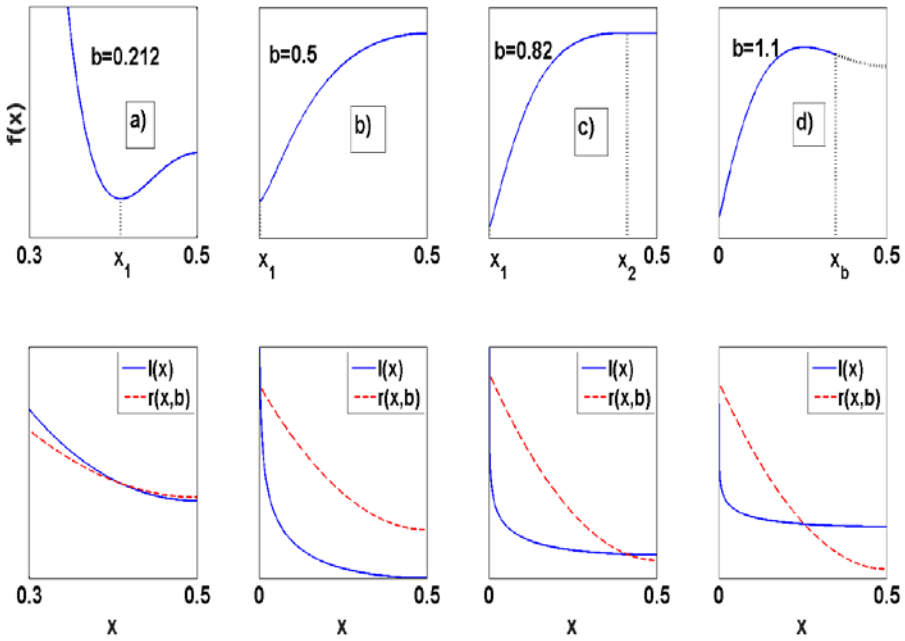


Рис.3. Графики функций $f(x)$ (верхние панели) и $l(x)$, $r(x,b)$ (нижние панели) для $q = 6$ и различных b.

Как только b превзойдет 1 правая граница интервала изменения x станет переменной – см. выражение (6) для x_b. Глобальный минимум $f(x_1)$ теперь надо сравнивать не с $f(x_0)$, а с много большим значением $f(x_b)$ - см. рис. 3d. При $b \to \infty$ интервал $(0, x_b)$ стягивается к началу координат, а точка глобального минимума x_1 стремится к 0. Так и должно происходить: когда температура стремится к 0, спиновая система стремится в основное состояние $\mathbf{s}_0 = (1,1,...,1)$.

2) Фазовый переход II рода. Пока значение параметра b не превосходит b_c (15), свободная энергия имеет вид (11). Когда b становится больше b_c, свободную энергию надо вычислять подстановкой $x_1(b)$ в (8). При $b = b_c$ происходит фазовый переход второго рода.

Подставим в выражение (15) значения $q = 6, 8, 10, 12, 14$, и вычислим критические значения обратной температуры для 3D-, 4D-, 5D-, 6D- и 7D-модели Изинга соответственно. Полученные числа помещены во вторую строку табл.1. В третьей строке таблицы приведены критические значения обратной температуры, полученные с помощью компьютерного моделирования [14-17]. Налицо хорошее согласие между нашей

оценкой и результатами компьютерного эксперимента.

Табл.1. Критические значения обратной температуры b_c для модели Изинга.

Размерность решетки	3D	4D	5D	6D	7D
Наша теория: формула (15)	0.2113	0.1464	0.1127	0.0918	0.0774
Компьютерное моделирование	0.2216	0.1489	0.1139	0.0923	0.0777

На рис.4 данные табл.1 представлены в графическом виде. Сплошной линией показаны результаты компьютерного моделирования, штриховой линией – наша оценка (15). Точечной линией показана оценка $b_c = 1/2D$, получающаяся в приближении среднего поля (уравнение Брэгга-Вильямса [1]). Видно, что наша формула описывает результаты компьютерного эксперимента много лучше. С ростом D согласие между нашей теорией и компьютерным экспериментом улучшается. Напомним, что параметр q может принимать любые значения, не только четные целочисленные – см. замечание в конце предыдущего раздела 2. Проводя стандартные вычисления можно получить значения критических показателей [1]. Для всех $q \geq 16/3$ они принимают классические значения, характерные для модели среднего поля: $\alpha = 0$, $\beta = 1/2$, $\gamma = \gamma' = 1$, $\delta = 3$. Для размерностей $D > 3$ это совпадает с тем, что было получено другими авторами [21, 22]. С другой стороны, для размерности $D = 3$ общепринятыми считаются неклассические значения критических показателей, приближенные значения которых получены методом ренорм-группы [18, 19].

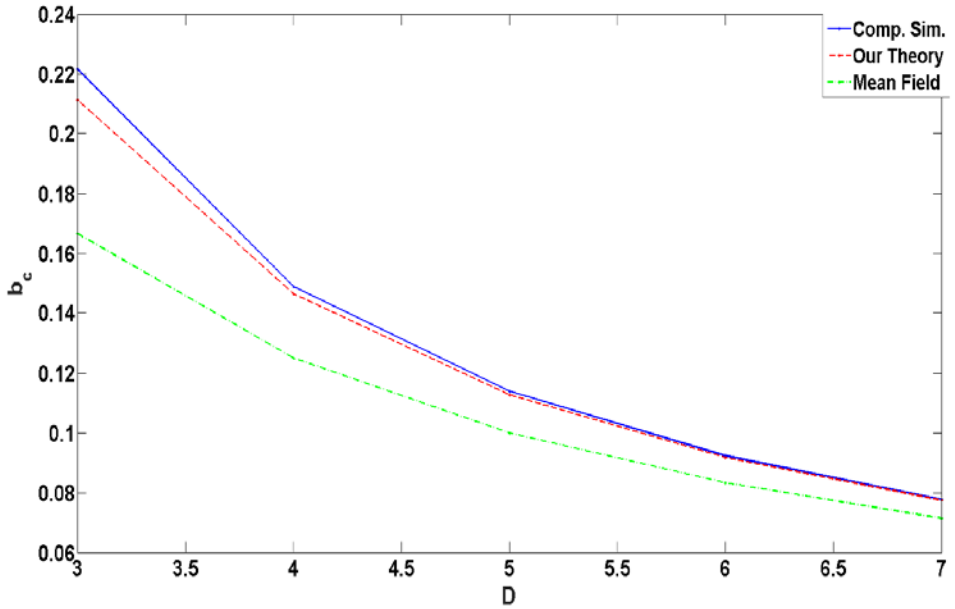


Рис.4. Зависимость b_c от размерности решетки D : компьютерное моделирование (сплошная линия), формула (15) (штриховая), модель среднего поля (точечная).

5. Интервал значений $q < 16/3$

1. Локальный минимум. Для $q < 16/3$, когда значение b станет больше b_t (13), у функции $f(x)$ появляются два дополнительных экстремума: точка минимума x_1 и точка максимума x_2: $x_1 < x_2$ - см. верхнюю панель рис.1d. По мере возрастания параметра b точка минимума x_1 будет сдвигаться по направлению к 0, а точка максимума x_2 устремится к правому концу интервала – к точке x_0. Необходимо следить за глубиной минимума $f(x_1)$: если в какой-то момент он станет глубже минимума $f(x_0)$, глобальный минимум функции $f(x)$ перескочит из точки x_0 в точку x_1. Такая коллизия описывается системой уравнений:

$$\begin{matrix} l(x) = r(x,b) \\ f(x) = f(x_0) \end{matrix}, \quad b \le 1. \qquad (17)$$

Условие $b \le 1$ означает, что внутренний локальный минимум $f(x_1)$ мы сравниваем с $f(x_0)$, а не со значением $f(x_b)$ на переменной правой границе x_b. Систему (17) удается решить подобно системе уравнений (12) - см. Приложение 3. Основной результат состоит в следующем.

Утверждение 2. Существует критическое значение параметра q: $q_c \approx 2.75$ такое, что для меньших значений q - $q < q_c$ - минимум $f(x_1)$ никогда не станет глубже $f(x_0)$.

Напротив, для любого q из интервала $q_c \le q < 16/3$ существует единственное меньшее 1 значение b_j, при котором локальный минимум в точке $x_1(b)$ становится равным глобальному минимуму в точке x_0: $f(x_1(b_j)) = f(x_0)$.

Как только b станет равным критическому значению b_j, глобальный минимум функции $f(x)$ перескочит из точки x_0 в точку $x_j = x_1(b_j)$ (индекс “j” от jump). Точка перескока x_j определяется пересечением прямой, параллельной оси абсцисс и отстоящей от нее на расстояние q, с графиком функции $G_1(x)$:

$$G_1(x) = \frac{\left[\xi_1(x) + \varphi_1(x)\right]^2}{\xi_1(x)} = q. \qquad (18)$$

Вспомогательные функции $\varphi_1(x)$ и $\xi_1(x)$ из уравнения (18) приведены в Приложении 3.

График функции $G_1(x)$ показан на рис. 5 сплошной линией. Значение b_j параметра b, при котором происходит перескок глобального минимума, вычисляется с помощью функции $B_1(x)$ и координаты x_j:

$$b_j = B_1(x_j), \text{ где } B_1(x) = \frac{\xi_1(x)}{\xi_1(x) + \varphi_1(x)}.$$

На рис.5 график функции $B_1(x)$ показан штриховой линией.

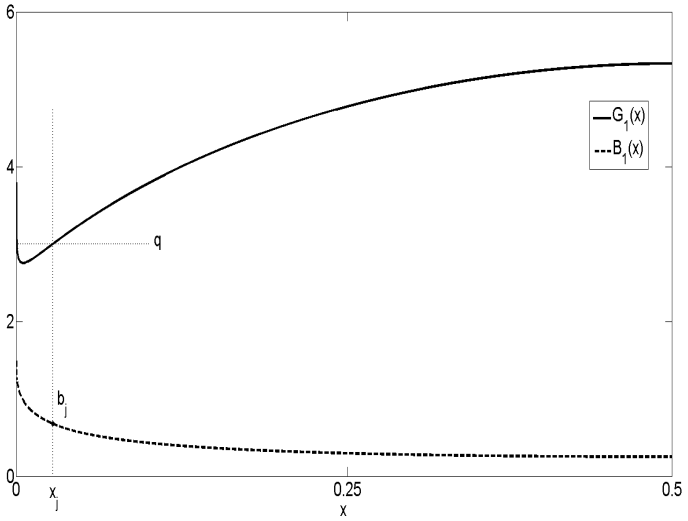


Рис.5. Графики функций $G_1(x)$, $B_1(x)$ для отыскания точки перескока глобального минимума.

Можно показать, что

$$\lim_{x\to 0} B_1(x) = 2,\ \lim_{x\to 0} G_1(x) \approx -\frac{\ln x}{4} \to +\infty,$$
$$\lim_{x\to 0.5} B_1(x) = 0.25, \quad \lim_{x\to 0.5} G_1(x) = 16/3.$$

Когда $x \to 0$ функции $G_1(x)$ и $B_1(x)$ логарифмически медленно стремятся к своим предельным значениям. С ростом x функция $B_1(x)$ монотонно убывает от $B_1(0) = 2$ до $B_1(0) = 0.25$. А вот функция $G_1(x)$ ведет себя немонотонно: поначалу она убывает от $+\infty$ до минимального значения $\min G_1(x) \approx 2.75$, причем $B_1(x)$ на этом интервале убывает от 2 до 1. Пройдя через минимум, функция $G_1(x)$ начинает монотонно возрастать, и на правом конце интервала стремится к значению $G_1(0.5) = 16/3$.

Из рис.5 видно, что при $q < q_c \approx 2.75$ уравнение (18) не имеет решения. Следовательно, для таких q локальный минимум $f(x_1)$ никогда не станет равным по глубине глобальному минимуму $f(x_0)$. Напротив, для $q_c \le q < 16/3$ уравнение (18) имеет в точности два решения (напомним, что при $x \to 0$ функция $G_1(x)$ стремится к $+\infty$, хотя и логарифмически медленно). Из двух решений нам подходит только большее - именно таким x_j отвечают значения параметра $b_j = B_1(x_1)$, меньшие 1: $b_j < 1$.

Полученные результаты можно использовать для анализа поведения спиновой системы.

2. Интервал значений $q < q_c \approx 2.75$. Для таких q локальный минимум в точке x_1 никогда не станет глубже минимума в точке x_0. Следовательно, пока b остается меньше 1, глобальный минимум $f(x)$ находится в точке x_0. Когда b превзойдет 1, и правая граница x_b (6) станет переменной, глобальный минимум функции $f(x)$ переместится в точку x_b. Выражение для свободной энергии в этом случае имеет вид:

$$\mathbf{f}(b) = b^{-1}\begin{cases} -\ln 2 - \dfrac{qb^2}{4}, & b \le 1 \\ L(x_b) - \dfrac{q}{4}(2b-1), & b \ge 1 \end{cases}. \qquad (19)$$

По мере неограниченного роста b глобальный минимум либо будет находиться в точке x_b, либо перескочит в текущее значение минимума x_1. В любом случае, с ростом b точка минимума стремится к 0, а система стремится в основное состояние.

Когда $b = 1$ первая производная свободной энергии (19) имеет разрыв:

$$\left.\frac{d\mathbf{f}}{db}\right|_{b=1-0} = -\frac{q}{2},\ \left.\frac{d\mathbf{f}}{db}\right|_{b=1+0} = \frac{1-q}{2},$$

что свидетельствует о фазовом переходе первого рода. Это справедливо для любого значения $q < q_c$, в частности, и для $q = 2$, отвечающему одномерной модели Изинга. С другой стороны, точное решение одномерной модели Изинга [1] не дает фазовых переходов при конечных температурах. Приходится констатировать, что в области $q < q_c$ наш метод дает неверные результаты.

3. Интервал значений $q_c \le q < 16/3$. Для таких q внутренний локальный минимум $f(x_1)$ при некотором значении b_j ($b_j < 1$) станет равным глобальному минимуму в точке x_0. В этот момент глобальный минимум функции $f(x)$ перескочит из x_0 в точку $x_j = x_1(b_j)$. Намагниченность состояния скачком изменится - с $m_0 = 0$ на $m_j = 1 - 2x_j$. По мере дальнейшего возрастания b точка глобального минимума $x_1(b)$ будет неуклонно стремиться к 0, а система – в основное состояние.

К данному интервалу значений q принадлежит $q = 4$, которое отвечает 2D модели Изинга. Получается, что наш подход предсказывает для 2D модели Изинга фазовый переход первого рода при $b_j \approx 0.3912$. Напротив, восходящее к Л. Онсагеру точное решение 2D модели Изинга дает фазовый переход второго рода при заметно большем критическом значении $b_c \approx 0.4407$ [1]. Для значений параметра q из этого интервала наш метод дает сомнительные результаты.

Проведенный анализ уравнения состояния подтверждается прямым вычислением таких физических характеристик как намагниченность, внутренняя энергия и свободная энергия – см. физический анализ полученных результатов в работе [20].

6. Обсуждение и выводы

Уточним сделанное у нас приближение. Оно состоит в том, что, при переходе от суммирования по n-окрестности Ω_n к

интегрированию, неизвестное нам истинное распределение энергий $P_n(E)$ заменяется гауссовой плотностью с известными средним значением E_x и дисперсией σ_x^2. Центральная часть распределения $P_n(E)$ аппроксимируется при этом очень хорошо, но хвосты $P_n(E)$ всегда не гауссовы ("толстые хвосты") [13]. Для нашего метода это и является основным источником ошибки. Покажем, что с ростом координационного числа q величина ошибки будет уменьшаться.

Перейдем в выражении (П2) к безразмерным нормированным величинам: $\varepsilon = E / |E_0|$, $\varepsilon_x = E_x / |E_0|$, $\tilde{\sigma}_x = \sigma_x / |E_0|$ и $\overline{\beta} = \hat{\beta} |E_0|$. Тогда выражение (П2) примет вид:

$$\frac{\sum_{s\in\Omega_n} e^{-\hat{\beta}E(s)}}{C_N^n} \approx \approx \frac{1}{\sqrt{2\pi}\tilde{\sigma}_x} \int_{-1}^{1} \exp\left[-\overline{\beta}\varepsilon - \frac{1}{2}\left(\frac{\varepsilon - \varepsilon_x}{\tilde{\sigma}_x} \right)^2 \right] d\varepsilon \quad . (20)$$

В новых обозначениях границы интегрирования не зависят от параметров задачи. То, насколько хорошо выполняется равенство (20), определяется только величиной стандартного отклонения $\tilde{\sigma}_x$: чем меньше $\tilde{\sigma}_x$, тем меньший вклад в интеграл в правой части (20) дают хвосты распределения, тем лучше гауссова плотность аппроксимирует распределение $P_n(E)$, и тем лучше выполняется равенство (20). Для модели Изинга имеем:

$$\tilde{\sigma}_x = 4x(1-x)\sqrt{\frac{2}{qN}}.$$

Как видим, с ростом координационного числа q величина $\tilde{\sigma}_x$ убывает. Соответственно, растет точность оценки интеграла (20) и точность получаемых в нашем подходе результатов.

Действительно, для решеток с большим значением координационного числа - $q > 16/3$ - метод n-окрестностей дает результаты, хорошо согласующиеся с имеющимися в литературе (раздел 4). Для спиновой системы предсказан фазовый переход второго рода, а аналитическое выражение (15) для критической температуры хорошо описывает результаты компьютерного эксперимента. Критические показатели принимают классические значения для всех таких q. Для размерностей $D \geq 4$ все это совпадает с результатами, полученными строгими методами [21, 22]. Для $D = 3$ метод ренорм-группы дает неклассические значения критических показателей.

К результатам, полученным для решеток с малым значением координационного числа $q_c < q < 16/3$, следует относиться критически. Для 2D модели Изинга ($q = 4$) метод n-окрестностей предсказывает скачок намагниченности при критической температуре, что противоречит известному точному решению задачи. Для малых значений координационного числа $q < q_c \approx 2.75$ метод n-окрестностей вовсе не работает.

Отсюда ясны границы применимости метода n-окрестностей для модели Изинга. Можно уверенно пользоваться методом для решеток больших размерностей $D \geq 3$. Для $D = 2$ можно численно рассчитать зависимость свободной энергии от обратной температуры - от истинной кривой она будет отличаться на 1-2 процента в районе критической температуры [20]. В современных методиках анализа сцен и компьютерной обработки изображений [23-26] многократно перевычисляется нормировочная константа, которая и есть значение статистической суммы при фиксированных значениях внешних параметров. Здесь востребованным оказывается быстрое – пусть даже и приближенное - вычисление статистической суммы, а критическая температура никого не интересует.

Данные рекомендации относятся к модели Изинга с взаимодействием между ближайшими соседями. Учет взаимодействия со следующими соседями приведет к увеличению эффективного координационного числа q. Это улучшит согласие между результатами, полученными методом n-окрестностей и точным решением (либо компьютерным экспериментом).

Работа проводилась в рамках выполнения проекта № 35.14 при финансовой поддержке грантами РФФИ №15-07-04861 и 16-01-00626.

Приложения.

Приложение 1.

Получим выражение для статистической суммы. В качестве начальной конфигурации возьмем основное состояние модели Изинга $\mathbf{s}_0 = (1,1,...,1)$. Тогда n-окрестность Ω_n $\mathbf{s}_0$ состоит из конфигураций с одним и тем же значением намагниченности:

$$m = \frac{1}{N}\sum_{i=1}^{N} s_i = 1 - \frac{2n}{N},\ n = 0,1,...,N\ .$$

Воспользуемся выражением (1) для энергии на один спин, и введем $\hat{\beta} = \beta N$, где $\beta = 1/T$ есть обратная температура. Для статистической суммы можно записать очевидные равенства:

$$Z_N = \sum_{\mathbf{s}} e^{-\hat{\beta}E(\mathbf{s},H)} = \sum_{n=0}^{N} e^{\hat{\beta}H\left(1-\frac{2n}{N}\right)} \sum_{\mathbf{s}\in\Omega_n} e^{-\hat{\beta}E(\mathbf{s})} = \\ = \sum_{n=0}^{N} C_N^n \cdot e^{\hat{\beta}H\left(1-\frac{2n}{N}\right)} \cdot \left(\frac{1}{C_N^n} \sum_{\mathbf{s}\in\Omega_n} e^{-\hat{\beta}E(\mathbf{s})} \right). \qquad (П1)$$

Устремим $N \to \infty$ и заменим в последнем выражении суммирование интегрированием по действительной переменной $x = n/N$: $x \in [0,1]$, воспользовавшись формулой Стирлинга: $C_N^{xN} \sim \exp(-N \cdot L(x))$, где $L(x) = x\ln x + (1-x)\ln(1-x)$. Теперь в (П1) суммирование по n можно заменить интегрированием по x: $\sum_{n=0}^{N} \to \int_0^1 dx$.

Усреднение по Ω_n заменим интегрированием по гауссовой плотности с известными средним значением E_x и дисперсией σ_x^2 (2), (3):

$$\frac{1}{C_N^n} \sum_{\mathbf{s}\in\Omega_n} e^{-\hat{\beta}E(\mathbf{s})} \approx \frac{1}{\sqrt{2\pi}\sigma_x} \int_{E_0}^{|E_0|} e^{-\hat{\beta}E - \frac{1}{2}\left(\frac{E-E_x}{\sigma_x}\right)^2} dE\ . \quad (П2)$$

Тогда можно записать:

$$Z_N \sim \int_0^1 \exp\left(-NL(x) + \hat{\beta}H(1-2x)\right)dx \times \\ \times \int_{E_0}^{|E_0|} \exp\left(-\hat{\beta}E - \frac{N}{2}\left(\frac{E-E_x}{\sqrt{N}\sigma_x}\right)^2 \right) dE \quad ,$$

где, используя выражения (2) и (3), имеем:

$$E_0 = -\frac{\sum_{ij} T_{ij}}{2N},\ E_x = E_0(1-2x)^2,\ \sigma_x^2 = \frac{8\sum_{ij} T_{ij}^2}{N^2} x^2(1-x)^2.$$

Если взаимодействие между ближайшими спинами равно J, то $E_0 = -DJ$, а $\sigma_x^2 = 16DJ^2x^2(1-x)^2/N$.

Вспоминая, что $\hat{\beta} = \beta N$, окончательно получаем:

$$Z_N \sim \int_0^1 dx \int_{E_0}^{|E_0|} \exp\left(-N \cdot F(x,E)\right)dE, \quad N >> 1,$$

$$F(x,E) = L(x) - \beta H(1-2x) + \beta E + \frac{1}{2N}\left(\frac{E-E_x}{\sigma_x}\right)^2.$$

Приложение 2.

Изучим условия, при которых кривые $l(x)$ и $r(x,b)$ касаются друг друга внутри интервала $(0, x_0)$. Распишем подробно систему уравнений (12):

$$\frac{\ln\frac{1-x}{x}}{2(1-2x)} = qb(1-b) + qb^2(1-2x)^2 \\ \frac{\ln\frac{1-x}{x} - \frac{1-2x}{2x(1-x)}}{(1-2x)^2} = -4qb^2(1-2x), \qquad x < 0.5\ .$$

Умножим второе уравнение на $(1-2x)/2$ и сложим его с первым уравнением, а само второе уравнение разделим на $(1-2x)^2$ и преобразуем к более удобному виду – получим новую систему:

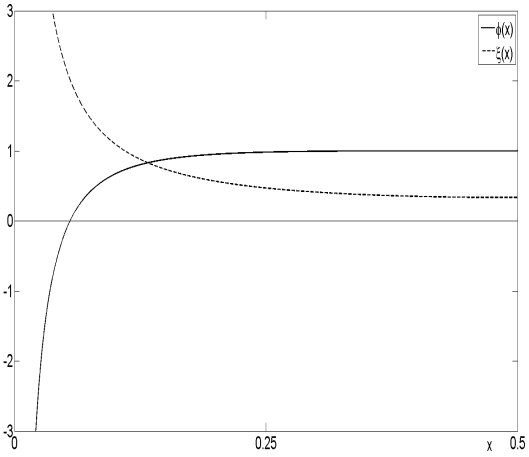


Рис.6. Графики функций $\varphi(x)$ и $\xi(x)$.

$$qb(1-b) = \frac{1}{2}\left(3\frac{\ln\frac{1-x}{x}}{2(1-2x)} - \frac{1}{4x(1-x)} \right) \\ qb^2 = \frac{1}{2(1-2x)^2}\left(\frac{1}{4x(1-x)} - \frac{\ln\frac{1-x}{x}}{2(1-2x)} \right) \qquad (П3)$$

Функции, стоящие в правой части системы (П3), обозначим через $\varphi(x)$ и $\xi(x)$ соответственно. С их помощью удобно записать окончательный ответ. Нетрудно видеть, что

$$\lim_{x\to 0}\varphi(x) = -\infty, \quad \lim_{x\to 0.5}\varphi(x) = 1, \\ \lim_{x\to 0}\xi(x) = +\infty, \ \lim_{x\to 0.5}\xi(x) = 1/3.$$

Графики функций $\varphi(x)$ и $\xi(x)$ см. на рис. 6.

Выразим b с помощью второго уравнения (П3) через q и x, и подставим это выражение в левую часть первого уравнения - получим уравнение, содержащее только x и q:

$$\varphi(x) = \sqrt{q}\cdot\sqrt{\xi(x)} - \xi(x)\ .$$

Получается, что точка касания кривых внутри интервала $(0, x_0)$ должна удовлетворять уравнению

$$G(x) = \frac{\left[\xi(x)+\varphi(x)\right]^2}{\xi(x)} = q\ . \quad (\text{П4})$$

Подставляя во второе уравнение системы (П3) вместо q левую часть выражения (П4), получаем значение b, при котором происходит касание кривых $l(x)$ и $r(x,b)$:

$$b = B(x) = \frac{\xi(x)}{\xi(x)+\varphi(x)}\ .$$

$G(x)$ - монотонно возрастающая функция (см. рис. 2). Нетрудно вычислить ее предел на правом конце интервала [0, 0.5]: $\lim\limits_{x\to 0.5} G(x) = 16/3$. Функция $B(x)$ монотонно убывает на интервале [0, 0.5]. Ее значение на правом конце интервала $\lim\limits_{x\to 0.5} B(x) = 0.25$.

Приложение 3.

Изучим условия, при которых происходит перескок глобального минимума из $x_0 = 0.5$ во внутреннюю точку интервала. Распишем систему уравнений (17):

$$\frac{\ln\dfrac{1-x}{x}}{2(1-2x)} = qb(1-b) + qb^2(1-2x)^2,$$

$$L(x) - \frac{qb}{2}\left[(1-2x)^2 + 8b\left(x(1-x)\right)^2\right] = \quad (\text{П5})$$

$$= -\ln 2 - \frac{qb^2}{4},$$

где $b \le 1$. Перегруппируем слагаемые в первом уравнении системы (П5), а второе уравнение преобразуем с помощью цепочки равенств:

$$L(x) + \ln 2 = \frac{qb}{4}(1-2x)^2\left[2 - b - 4bx(1-x)\right] =$$

$$= \frac{qb}{4}(1-2x)^2\left[1 - b + \frac{\ln\dfrac{1-x}{x}}{2(1-2x)}\right].$$

Получим новую систему уравнений:

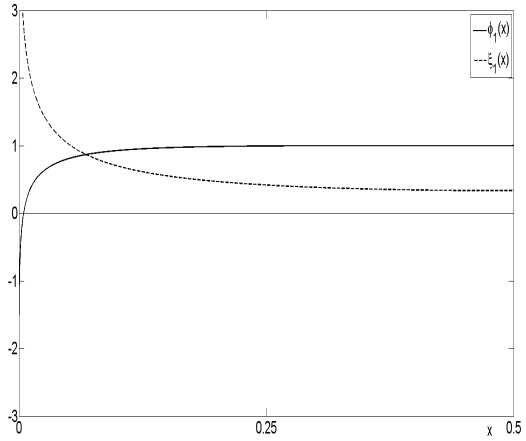


Рис.7. Графики функций $\varphi_1(x)$ и $\xi_1(x)$.

$$qb^2 = \frac{2}{(1-2x)^2}\left[\frac{\ln\dfrac{1-x}{x}}{2(1-2x)} - \frac{2\left(L(x)+\ln 2\right)}{(1-2x)^2}\right] \quad (\text{П6})$$

$$qb(1-b) = 4\frac{L(x)+\ln 2}{(1-2x)^2} - \frac{\ln\dfrac{1-x}{x}}{2(1-2x)}.$$

Правые части полученных выражений обозначены через $\varphi_1(x)$ и $\xi_1(x)$. Эти функции ведут себя аналогично функциям $\varphi(x)$ и $\xi(x)$ из (П3), их графики представлены на рис. 7.

Нетрудно вычислить предельные значения для этих функций:

$$\lim_{x\to 0}\varphi_1(x) = -\infty,\ \lim_{x\to 0.5}\varphi_1(x) = 1,$$

$$\lim_{x\to 0}\xi_1(x) = \infty,\ \lim_{x\to 0.5}\xi_1(x) = \frac{1}{3}.$$

Отметим логарифмически медленное стремление функций к бесконечности, когда $x \to 0$.

Выражая из первого уравнения (П6) b через q и x, и подставляя это выражение в левую часть второго уравнения, получаем уравнение для отыскания абсциссы, для которой локальный минимум сравняется по глубине с глобальным минимумом $f(x_0)$:

$$G_1(x) = \frac{\left[\xi_1(x)+\varphi_1(x)\right]^2}{\xi_1(x)} = q\ . \quad (\text{П7})$$

График функции $G_1(x)$ представлен на рис.5. Локальный минимум сравняется по глубине с $f(x_0)$ в точке x_j, являющейся решением уравнения (П7): $G_1(x_j) = q$.

Подставляя в первое уравнение системы (П6) вместо q левую часть выражения (П7),

получаем значение b, при котором оба минимума станут равными по глубине:

$$b = B_1(x) = \frac{\xi_1(x)}{\xi_1(x) + \varphi_1(x)}.$$

Значение параметра b, при котором происходит перескок глобального минимума из x_0 в x_j есть $b_j = B_1(x_j)$. График функции $B_1(x)$ представлен на рис. 5.

Applicability of n-vicinity method for investigation of Ising model

B. Kryzhanovsky, Лю Litinskii

Abstract. Here we apply the n-vicinity method of approximate calculation of the partition function to an Ising Model with the nearest neighbor interaction on D-dimensional hypercube lattice. We solve the equation of state for an arbitrary dimension D and analyze the behavior of the free energy. As expected, for large dimensions ($D \geq 3$) the system demonstrates a phase transition of the second kind. In this case, we obtain a simple analytical expression for the critical value of the inverse temperature. When $3 \leq D \leq 7$ this expression is in a very good agreement with the results of computer simulations. In the case of small dimensions ($D = 1, 2$), there is a noticeable discrepancy with the known exact results.

Keywords: the n-vicinity method; the Ising model; critical temperature.

Литература

1. Р. Бэкстер. Точно решаемые модели в статистической механике. М., Мир, 1985.
2. M. Kac and J. Ward. A combinatorial solution of the two-dimensional Ising model. «Phys. Rev.», vol.88 (1952), p. 6.
3. P. Kasteleyn. Dimer statistics and phase transitions. «Journal of Math. Phys.», vol. 4 (1963), p. 2.
4. D. Amit, H. Gutfreund and H. Sompolinsky. Statistical Mechanics of Neural Networks Near Saturation. «Annals of Physics», vol.173 (1987), p. 30-67.
5. A.K. Hartmann and H. Rieger (Eds.). New optimization algorithms in physics (Chapter III). Berlin, Wiley-VCH, 2004.
6. J.M. Dixon, J.A. Tuszynski, E.J. Carpenter. Analytical expressions for energies, degeneracies and critical temperatures of the 2D square and 3D cubic Ising models. «Physica A», vol.349 (2005), pp. 487 – 510, doi:10.1016/j.physa.2004.10.029.
7. O.C. Martin, R. Monasson, R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems. «Theoretical Computer Science», vol.265(1-2) (2001), pp. 3-67, http://arxiv.org/abs/cond-mat/0104428.
8. G.E. Hinton, S. Osindero, Y. The. A fast learning algorithm for deep belief nets. «Neural Computation», vol. 18 (2006), pp.1527-1554.
9. M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. «IEEE Trans. on Information Theory», vol. 51 (2005), №7, pp.2313–2335.
10. J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. «IEEE Trans. on Information Theory», vol. 5 (2005), pp.2282–2312.
11. B. Kryzhanovsky, L. Litinskii. Approximate method of free energy calculation for spin system with arbitrary connection matrix. «Journal of Physics: Conference Series», vol. 574 (2015), p. 012017. Preprint arXiv: 1410.6696.
12. Б.В. Крыжановский, Л.Б. Литинский. Обобщенное уравнение Брэгга-Вильямса для систем с произвольным дальнодействием. «ДАН», т.459 (2014), №6, стр. 1-5.
13. Boris Kryzhanovsky and Leonid Litinskii. Generalized approach to description of energy distribution of spin system. «Optical Memory & Neural Networks (Information Optics)», v.24 (2015) , No.3, p. 165-185. Preprint arXiv: 1505.03393.
14. H.W.J. Blote, L.N. Shchur, A.L. Talapov. The cluster processor: new results. «Int. J. Mod. Phys. C.», vol. 10 (1999), p.1137.
15. R. Ндggkvist, A. Rosengren, PH Lundow, K. Markstrцm, D. Andren, P. Kundrotas. On the Ising model for the simple cubic lattice. «Advances in Physics», vol. 56 (2007), №5, pp. 653-755.

16. P.H. Lundow, K. Markstrom. The critical behaviour of the Ising model on the 4-dimensional lattice. «Phys. Rev. E», vol. 80 (2009), p.031104. Preprint arXiv:1202.3031v1.
17. P.H. Lundow, K. Markstrom. The discontinuity of the specific heat for the 5D Ising model. «Nuclear Physics B», v. 895 (2015), p.305.
18. R. Guida, J. Zinn-Justin. Critical exponents of the N-vector model. «Phys. A: Math. Gen.», vol.31 (1998), p. 8103.
19. Р. Фольк, Ю. Головач, Т. Яворский. Критические показатели трехмерной слабо разбавленной замороженной модели Изинга. «УФН», т.173 (2003), № 2, стр. 175– 200.
20. Б.В. Крыжановский, Л.Б. Литинский. Приближенное вычисление статистической суммы для модели Изинга на гиперкубе. «Труда НИИСИ РАН», т.6 (2016), №2, стр. 5-10.
21. A. Sakai. Lace expansion for the Ising Model. «Commun Math. Phys.», vol. 272 (2007), pp. 283-344.
22. M. Heydenreich, R. van der Hofstad, A. Sakai. Mean-Field Behavior for Long- and Finite Range Ising Model Percolation and Self-Avoiding Walk. «J. Stat. Phys.», vol. 132 (2008), pp. 1001-1049.
23. C. Wang, N. Komodakis, N. Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. «Preprint to Elsevier» (2013).
24. M.P.C. Fossorier, M. Mihaljevicм, H. Imai. Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation. «IEEE Transactions on Communications», vol. 47 (1999), № 5, pp. 673-680.
25. J. Liu, R. De Lamare. Low-latency reweighted belief propagation decoding for LDPC codes. «IEEE Communications Letters», 2012.
26. A. Novikov, A. Rodomanov, A. Osokin, D. Vetrov. Putting MRFs on a Tensor Train. (Proc. of the 31st Intern. Conf. on Machine Learning, Beijing, China, 2014). «Journal of Machine Learning Research: W&CP», vol. 32.

Полиномиальный алгоритм точного вычисления статистической суммы для модели бинарных спинов на планарных графах

Я.М. Карандашев[1], М.Ю. Мальсагов[2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail's: [1] karandashev@niisi.ras.ru, [2] malsagov@niisi.ras.ru

Аннотация: В данной работе предложен и реализован (код доступен по ссылке https://github.com/Thrawn1985/2D-Partition-Function) алгоритм точного вычисления статистической суммы для двумерных графических моделей с бинарными переменными. Сложность алгоритма составляет *O*(*N*^2). Тестовые эксперименты показали хорошее согласие с аналитическим решением Онсагера для двумерной модели Изинга.

Ключевые слова: Планарный граф, модель Изинга, статистическая сумма, бинарная модель, полиномиальный алгоритм.

1. Введение

В настоящей работе описывается алгоритм решения задачи поиска статистической суммы системы взаимодействующих спинов в узлах решётки, когда спины принимают два значения (+/-1), а энергия их взаимодействия описывается квадратичной функцией, как в модели Изинга.

Вычисление статистической суммы играет важную роль в физике, химии, в области компьютерного зрения и машинного обучения. Применение графических моделей обычно включает в себя вычисление двух величин: оценку апостериорного максимума вероятности и маргинальных распределений. Вычисление последних тесно связано с подсчётом статической суммы [1]. Задача состоит в том, чтобы вывести некоторые статистические свойства (например, маргинальные вероятности) при заданном наборе случайных переменных в известной графической модели.

Известно, что проблема вычисления статистической суммы решается точно лишь для очень ограниченного числа задач. В частности, это задачи на планарных графах (двумерных решётках), на цепочках или графах небольшого размера. В большинстве случаев, когда задача имеет размерность порядка нескольких сотен или тысяч – задача становится трудно решаемой, что объясняется её экспоненциальной сложностью. В этом случае приходится ограничиться эвристическими приближёнными методами. Для развития эвристических методов приближённого вычисления статсуммы удобно иметь в распоряжении и точные методы, хотя бы для узкого класса задач.

В работах [2, 3] предложили приближённый метод, основанный на деревьях, известный как tree reweighting (TRW). Алгоритм TRW предназначен для решения задачи приближенного нахождения наиболее вероятной конфигурации скрытых переменных в циклических графических моделях общего вида.

В работе [4] рассмотрен другой класс разрешимых моделей: планарные графы. Граф называют планарным, если его можно нарисовать на плоскости без пересечения рёбер. Было обнаружено, что в особом случае модели Изинга [5] со спинами {−1, +1} и попарным взаимодействием, определённом на планарном графе, вычисление статистической суммы полиномиально сводится к задаче вычисления детерминанта матрицы [6-9]. Реализации этих методов посвящена настоящая работа. Интересно отметить, что с 60-х годов не было предложено существенно более совершенных алгоритмов. Эти же алгоритмы используются в машинном обучении [10, 11].

Статья организована следующим образом. В главе 2 описывается предложенный алгоритм, основанный на работах Кастелейна и Фишера. В главе 3 приводятся результаты тестовых экспериментов, оценки временной сложности, а также сравнение с

аналитическим решением Онсагера в пределе больших размерностей.

2. Описание алгоритма

Метод подсчёта статистической суммы на планарном графе сводится к подсчёту количества идеальных паросочетаний ребер графа с учётом их веса, используя методы из линейной алгебры, как было предложено в 1961 году Кастелейном и Фишером. Некоторые части алгоритма были взяты из книги [12], а также из работ [13, 14].

Пусть задан планарный граф G, для которого нужно посчитать статистическую сумму. Вкратце, алгоритм состоит в следующем:

1. Для исходного планарного графа G строится дуальный граф D.
2. Вершины графа D, имеющие степень больше 2, определённым образом разворачиваются с сохранением свойства планарности, приводя к расширенному графу R с матрицей связей $W = \{w_{ij}\}$.
3. Строится кососимметричная матрица $B = \{b_{ij}\}$, соответствующая Пфаффовой ориентации графа R.
4. Искомая статистическая сумма равна Пфаффиану матрицы $A = \{a_{ij} = b_{ij} w_{ij}\}$, который равен квадратному корню из детерминанта, т.е. $Z = \mathrm{Pf}(A) = \sqrt{\det A}$.

Пункты 3-4 известны в англоязычной литературе как алгоритм FKT (Fisher-Kasteleyn-Temperley algorithm). Каждый из 4 шагов алгоритма известен, но на данный момент, во-первых, не было подробного совместного описания всех четырёх пунктов, решающих задачу вычисления статистической суммы на планарных графах, а во-вторых, не было их реализации, кроме [10].

Ниже каждый из четырёх пунктов разобран подробнее.

2.1. Построение дуального графа

Пусть есть исходный граф G с матрицей связи J_{ij}. Пусть в каждой вершине расположена частица со спином, который может принимать два значения $s_i = \pm 1$, которые мы будем принимать за состояние вершины. Требуется вычислить статсумму по всем возможным состояниям вершин графа:

$$Z = \sum_s e^{-\beta E(s)}, \qquad (1)$$

где β - обратная температура, а энергия определяется попарными взаимодействиями между вершинами графа:

$$E(s) = -\sum_{i,j}^{N} J_{ij} s_i s_j . \qquad (2)$$

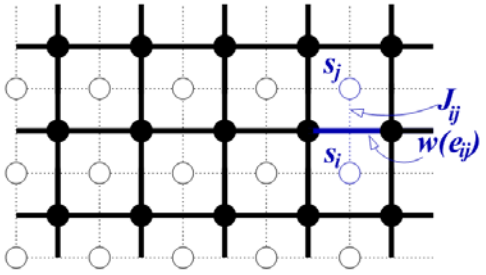


Рис. 1. Дуальный граф обозначен жирными линиями и заполненными кружками, исходный граф – тонкими линиями и белыми кружками.

Пусть $V = \{i : s_i = +1\}$, т.е. V - множество вершин, для которых состояние +1. Тогда выражение для энергии можно переписать следующим образом:

$$\begin{aligned} E(s) &= -\sum_{i,j}^{N} J_{ij} s_i s_j = \\ &= 2\sum_{i\in V}\sum_{j\notin V} J_{ij} - \sum_{i,j\in V}^{N} J_{ij} - \sum_{i,j\notin V}^{N} J_{ij} = \\ &= 4\sum_{i\in V}\sum_{j\notin V} J_{ij} - \sum_{i,j}^{N} J_{ij} \end{aligned} \qquad (3)$$

Видно, что второй член $\sum_{i,j}^{N} J_{ij}$ - константа, а первый член $\sum_{i\in V}\sum_{j\notin V} J_{ij}$ определяется лишь парами вершин $<i, j>$, такими что i-й элемент имеет одно состояние, а j-й элемент – противоположное. Особенно наглядно это видно на планарном графе, где такие пары вершин лежат на границах множества V.

Для случая планарного графа G, который здесь рассматривается, можно построить дуальный (или двойственный) граф D, вершинами которого являются грани (*faces*) исходного графа, а веса рёбер определим следующим образом (см. рис. 1):

$$w_{ij} = e^{-4\beta J_{ij}} . \qquad (4)$$

Для дуального графа можно заметить следующее (рис. 2): любой конфигурации состояний вершин s в исходном графе G (рис. 2а) однозначно соответствует набор эйлеровых циклов в дуальном графе D (рис. 2б), т.е. замкнутых кривых, проходящих по рёбрам w_{ij} и ограничивающих вершины s_i в состоянии +1. Таким образом, нахождение статсуммы (1) эквивалентно суммированию по

всевозможным эйлеровым подграфам в дуальном графе D, т.е.:

$$Z = C\sum_{V} e^{-4\beta\sum_{i\in V}\sum_{j\notin V} J_{ij}} = C \sum_{\substack{\varnothing-\text{эйлеровы} \\ \text{подграфы в } D}} \prod_{e\in\varnothing} w_e , \quad (5)$$

где

$$C = \exp\left(\beta\sum_{i,j}^{N} J_{ij}\right) \quad (6)$$

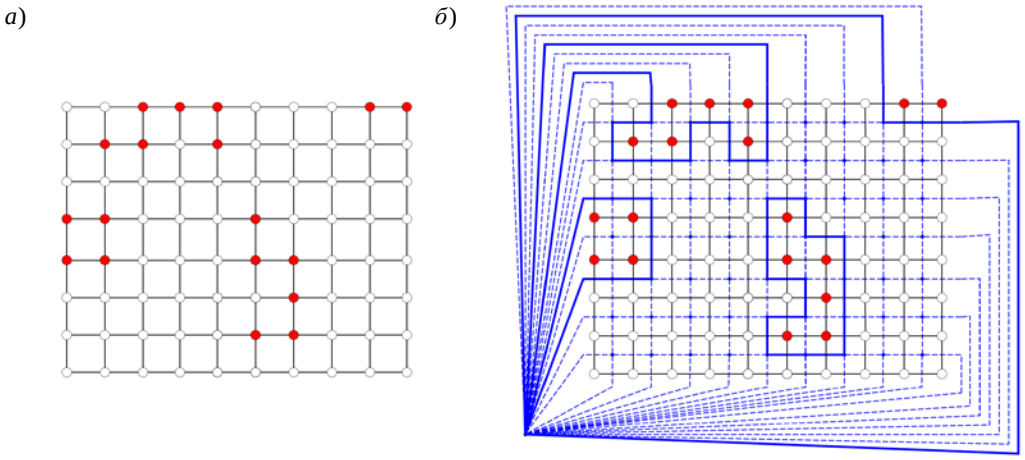


Рис. 2. а) Исходная конфигурация, где заполненные кружки соответствуют состоянию +1, а пустые -1. б) Дуальный граф (обозначен пунктирными линиями и маленькими точками). Жирным выделены эйлеровы циклы, соответствующие исходной конфигурации.

2.2. Развёртывание вершин

Пусть дуальный граф D построен. Сделаем следующие две операции:

1. Во-первых, все вершины графа D, имеющие степень больше 3, развернем, как показано на рис. 3. В итоге, все вершины будут иметь степень 3.
2. На втором шаге каждую вершину (имеющую уже степень равную 3) заменим как на рис. 4. Такая розетка, как на рис. 4б, выбрана потому, что она всегда допускает идеальное паросочетание.

В итоге, после такого преобразования, получим новый граф, назовём его граф R. Особенность этого графа в том, что он, во-первых, по-прежнему остаётся планарным. Во-вторых, количество вершин в нём чётное, и в нём всегда найдётся идеальное паросочетание, при этом можно показать, что каждому идеальному паросочетанию в графе R после обратного сжатия вершин будет соответствовать подграф из эйлеровых циклов в графе D.

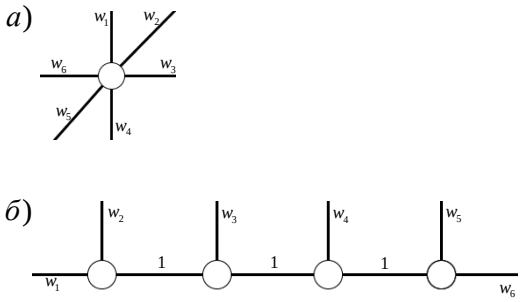


Рис. 3. Разворачивание вершин, шаг первый. Вершина степени больше 3, показанная на рис. а) заменяется несколькими вершинами, как на рис. б).

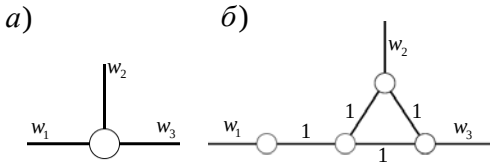


Рис. 4. Разворачивание вершин, шаг второй. Вершина, показанная на рис. а) заменяется 4-мя вершинами, как на б).

2.3. Идеальные паросочетания и поиск пфаффовой ориентации в графе

Известно, что для любой кососимметрической матрицы $A = (a_{ij} : a_{ij} = -a_{ji})$ размера $2n \times 2n$ можно определить пфаффиан по формуле:

$$\mathrm{pf}(A)=\sum_{\pi} sign\begin{pmatrix}1 & 2 & \dots & 2n\\ i_1 & j_1 & \dots & j_n\end{pmatrix} a_{i_1 j_1} a_{i_2 j_2} \dots a_{i_n j_n}\ , \quad (7)$$

где суммирование ведётся по всем разбиениям $\pi=\{\{i_1,j_1\},\dots,\{i_n,j_n\}\}$ множества {1,…,2n} на пары, *sign* – означает знак подстановки $\begin{pmatrix}1 & 2 & \dots & 2n\\ i_1 & j_1 & \dots & j_n\end{pmatrix}$.

До сих пор граф *R* был неориентированным. Допустим, что мы выбрали в графе *R* некоторую произвольную ориентацию рёбер. Обозначим такой орграф через $\vec{R}$ и определим матрицу *A* следующим образом:

$$a_{ij}=b_{ij}w_{ij},\ \text{где}\ b_{ij}=\begin{cases}1, & \text{если } (i,j)\in e(\vec{R})\\ -1, & \text{если } (j,i)\in e(\vec{R})\\ 0, & \text{иначе}\end{cases} \quad (8)$$

Заметим, что для графа $\vec{R}$ каждому члену суммы (7), относящемуся к разбиению на пары $\pi=\{\{i_1,j_1\},\dots,\{i_n,j_n\}\}$, соответствует некоторое паросочетание. Если какое-то одно ребро в паросочетании отсутствует (т.е. его вес равен $a_{i_k j_k}=0$), то всё произведение $a_{i_1 j_1} a_{i_2 j_2} \dots a_{i_n j_n}$ обнуляется. Таким образом, суммирование в (7) эквивалентно суммированию лишь по существующим идеальным паросочетаниям графа $\vec{R}$. А поскольку, как было сказано в предыдущем пункте, каждому идеальному паросочетанию в графе *R* соответствует набор эйлеровых циклов в графе *D*, то статсумму можно посчитать, перебрав все идеальные паросочетания:

$$Z=C\sum_{\substack{\varnothing-\text{Эйлеровы}\\ \text{подграфы в } D}}\prod_{e\in\varnothing} w_e=$$
$$=C\sum_{\substack{PM-\text{идеальные}\\ \text{паросочетания}\\ \text{в } \vec{R}}}\prod_{e\in PM} w_e \ge C\cdot\left|\mathrm{Pf}(A)\right| \quad (9)$$

Неравенство в правой части вызвано тем, что суммирование в пфаффиане (7) производится с различными знаками. С другой стороны, если подобрать такую ориентацию рёбер в графе $\vec{R}$, чтобы матрица $B=\{b_{ij}\}$ компенсировала знаки перестановок π, то все ненулевые слагаемые в сумме (7) шли бы с плюсом и таким образом мы бы получили:

$$Z=C\cdot\mathrm{Pf}(A). \quad (10)$$

Соответствующая ориентация $B=\{b_{ij}\}$ называется пфаффовой и, что самое главное, согласно теореме Кастелейна, для планарных графов она действительно существует и её можно найти за полиномиальное время.

Критерием проверки пфаффовой ориентации является теорема:

Если $\vec{R}$ - связный плоский ориентированный граф, у которого граница каждой грани (за исключением, возможно, бесконечной грани) имеет нечётное число рёбер, ориентированных по часовой стрелке, то орграф $\vec{R}$ является пфаффовым.

Алгоритм нахождения Пфаффовой ориентации следующий (рис. 5). Мы применим индукцию по числу рёбер в графе *R*. Если граф *R* является деревом, то подойдёт любая ориентация. Теперь предположим, что он не является деревом, и выберем произвольное принадлежащее циклу ребро, лежащее на границе бесконечной грани. Пусть F_0 – конечная грань, содержащая это ребро *e*.

В силу индуктивного предположения, граф (*R – e)* имеет такую ориентацию, при которой каждая конечная грань содержит в своей границе нечётное число рёбер, ориентированных по часовой стрелке. Вернём ребро *e* в граф и припишем ему такую ориентацию, чтобы граница грани F_0 имела нечётное число рёбер, ориентированных по часовой стрелке. Поскольку все границы конечных граней, отличных от F_0, не изменялись, то полученная ориентация графа *R* будет обладать нужными свойствами.

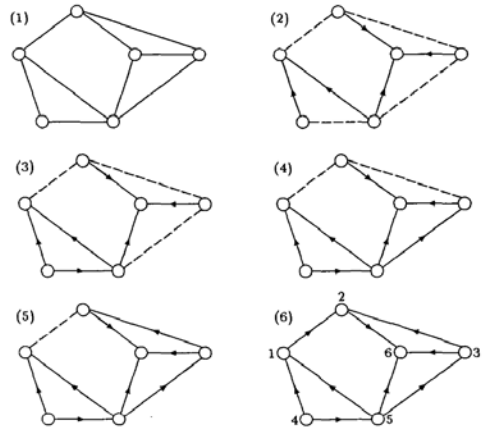


Рис. 5. Построение Пфаффовой ориентации планарного графа. 1. Исходный неориентированный граф. 2-6. Пошаговое построение ориентации.

2.4. Вычисление Пфаффиана и статсуммы

После того, как найдена Пфаффова ориентация графа $\vec{R}$, определяется матрица *A* по формуле (8), и нахождение статсуммы не составляет труда:

$$Z=C\cdot\mathrm{Pf}(A)=C\cdot\sqrt{\det(A)} \quad (11)$$

поскольку как известно квадрат Пфаффиана кососимметричной матрице равен детерминанту этой матрицы.

3. Экспериментальные результаты

Алгоритм, описанный в главе 2, был реализован[1] и протестирован на квадратных решётках со свободными граничными условиями и единичными весами связей.

Введём некоторые определения. Свободная энергия определяется как:

$$F = -kT\frac{\ln Z}{N} = -\frac{\ln Z}{\beta N} \qquad (12)$$

Удобнее пользоваться величиной $f = \beta F$, которую можно назвать «нормированной свободной энергией» - в ее терминах проще выражения для внутренней энергии и теплоемкости:

$$U = \frac{\partial f}{\partial \beta}, \quad C = -\frac{\partial^2 f}{\partial \beta^2} \qquad (13)$$

В экспериментах мы сравнили полученные с помощью нашего алгоритма значения f, а также её 1-й и 2-й производных (13) с аналитическими выражениями, полученными Онсагером для случая, когда размерность стремится к бесконечности. Мы рассмотрели квадратные решётки размером $N = n \times n$ для $n = 20, 50, 100$ и 200.

На рис. 6 показан график зависимости времени работы алгоритма от размерности. Для наглядности в качестве оси ординат показано не само время, а квадратный корень от него. Таким образом, получаем квадратичную сложность алгоритма $O(N^2)$ или четвертую степень $O(n^4)$ от линейного размера решётки.

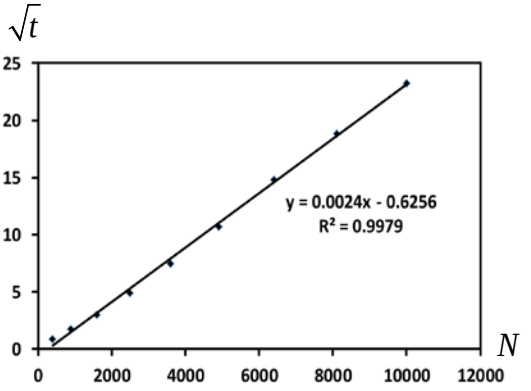


Рис. 6. Время работы алгоритма. Время t мерялось в секундах.

[1] Код доступен по ссылке https://github.com/Thrawn1985/2D-Partition-Function

Решение Онсагера [5] выглядит следующим образом:

$$f = \lim_{N\to\infty}\frac{Z}{N} = \frac{\ln 2}{2} + \frac{1}{2\pi}\int_0^{\pi} F d\theta,$$

$$F = \ln\left[\operatorname{ch}(4\beta J)\operatorname{ch}(4\beta K) + \frac{\sqrt{1+\chi^2-2\chi\cos 2\theta}}{\chi}\right] \qquad (12)$$

где

$$\chi = \frac{1}{\operatorname{sh}(4\beta J)\operatorname{sh}(4\beta K)} \qquad (13)$$

J и K - веса вертикальных и горизонтальных связей в решётке. В случае когда $K = J$, выражение для свободной энергии упрощается:

$$f = \lim_{N\to\infty}\frac{Z}{N} = \frac{\ln 2}{2} + \ln\left(\operatorname{ch}(4\beta J)\right) + \frac{1}{2\pi}\int_0^{\pi}\ln\left[1+\sqrt{1-\lambda^2\cos^2\theta}\right]d\theta \qquad (14)$$

где

$$\lambda = \frac{2\operatorname{sh}(4\beta J)}{\operatorname{ch}^2(4\beta J)} \qquad (15)$$

Значения свободной энергии, внутренней энергии и теплоёмкости показаны на рис. 7-9. Для подсчёта первой и второй производных в экспериментах мы вычисляли три значения свободной энергии в трёх близких точках отстоящих на $d\beta = 10^{-5}$ и вычисляли производные методом конечных разностей.

Как видно из графиков на рис. 7-9, экспериментально полученные значения свободной энергии и её производных при увеличении размерности приближаются к предельному решению Онсагера.

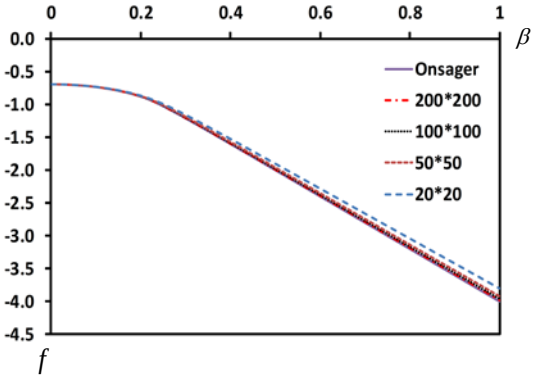


Рис. 7. График свободной энергии $f(\beta)$ для нескольких значений размеров решётки и решение Онсагера для случая бесконечной размерности.

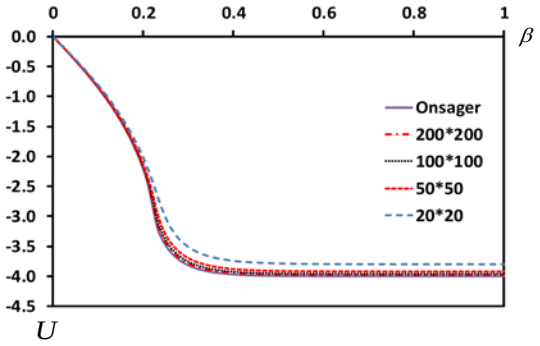


Рис. 8. График внутренней энергии $U(\beta)$ для нескольких значений размеров решётки и решение Онсагера для случая бесконечной размерности.

При больших значениях β значение статсуммы определяется главным образом энергией глобального минимума равной:

$$E_0 = -4n(n-1), \qquad (16)$$

поэтому значение свободной энергии линейно зависит от β как:

$$f = -4\beta\left(1-\frac{1}{n}\right). \qquad (17)$$

Рис. 7-8 подтверждают, что чем больше размерность, тем ближе наклон прямой к предельному значению -4.

Соответственно, при больших значениях β, значение 2-ой производной стремится к нулю. При критическом значении $\beta_{cr} \approx 0.22$, согласно решению Онсагера (14), теплоёмкость претерпевает разрыв и уходит в бесконечность. Для конечных размерностей такого не происходит, однако пик всё равно наблюдается - тем острее, чем больше размерность решётки (рис. 9).

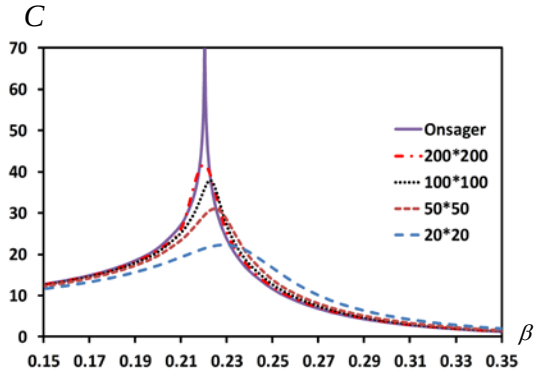


Рис. 9. График теплоёмкости для нескольких значений размеров решётки и решение Онсагера для случая бесконечной размерности. Для наглядности шкала по оси β увеличена в сравнении с рис. 7 и 8.

Финальный эксперимент, который был проведён в данной работе, направлен на исследование перехода от двумерной модели Изинга к одномерной. В эксперименте вес горизонтальных связей K плавно менялся от 1 до 0 (см. рис. 10) при фиксированном значении веса вертикальных связей $J = 1$. Как видно из рис. 10, положение пиков сдвигается вправо, что соответствует увеличению критической температуры β_{cr}. При $K = 0$ кривая становится монотонной, пик исчезает, двумерная решётка превращается в набор одномерных цепочек, для которых, как известно, фазовый переход происходит при $\beta = \infty$.

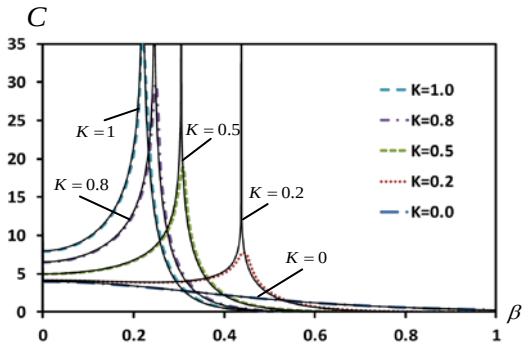


Рис. 10. Теплоёмкость для различных значений весов горизонтальных связей K от 0 до 1 при фиксированном значении вертикальных связей $J = 1$. Сплошные кривые показывают решение Онсагера, разрывные кривые получены экспериментально для $N = 100$.

4. Заключение

В данной работе предложен и реализован алгоритм точного вычисления статистической суммы для двумерных графических моделей с бинарными переменными. Алгоритм реализован на языке C/C++ с использованием библиотек Boost и Csparse [15]. Сложность алгоритма составляет $O(N^2)$ и ограничивается главным образом временем вычисления детерминанта разреженной матрицы.

Дальнейшие работы будут направлены на разработку эвристических алгоритмов для графов более общего вида. Данный алгоритм может быть использован для сравнения полученных приближённых результатов с точными.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект 16-31-00047) и проекта РАН 35.14.

Polynomial algorithm for exact calculation of partition function for binary spin model on planar graphs

I. Karandashev, M. Malsagov

Abstract: In this paper we propose and realize (the code is publicly available at https://github.com/Thrawn1985/2D-Partition-Function) an algorithm for exact calculation of partition function for planar graph models with binary spins. The complexity of the algorithm is $O(N^2)$. Test experiments shows good agreement with Onsager's analytical solution for two-dimensional Ising model of infinite size.

Keywords: Planar graph, Ising model, partition function, binary model, polynomial algorithm,

Литература

1. M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. Technical report, UC Berkeley Dept. of Statistics, 2003.
2. M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. IEEE Trans. on Information Theory, 49(5):1120–1146, 2003.
3. M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. IEEE Trans. on Information Theory, 51(7):2313–2335, 2005.
4. A. Globerson and T. Jaakkola. Approximate inference using planar graph decomposition. In NIPS, 2007.
5. L. Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. Phys. Rev., 65(3-4), 1944
6. M. Kac and J. Ward. A combinatorial solution of the two-dimensional Ising model. Phys. Rev., 88(6), 1952.
7. S. Sherman. Combinatorial aspects of the Ising model for ferromagnetism. i. a conjecture of Feynman on paths and graphs. J. Math. Phys., 1(3), 1960.
8. P. Kasteleyn. Dimer statistics and phase transitions. J.Math. Phys., 4(2), 1963.
9. M. Fisher. On the dimer solution of planar Ising models. J. Math. Phys., 7(10), 1966.
10. N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar Ising models. In NIPS, 2008.
11. J. Johnson, D. Oyen, M. Chertkov, P. Netrapalli. Learning Planar Ising Models, arXiv:1502.00916, 2015.
12. Л.Ловас, М.Пламмер. Прикладные задачи теории графов. Теория паросочетаний в математике, физике, химии. - М.: Мир, 1998. - 653 с.
13. A.Middleton, T.Middleton, K. Creighton. "Matching Kasteleyn Cities for Spin Glass Ground States" (2007). *Physics*. Paper 180. http://surface.syr.edu/phy/180
14. F. Liers and G. Pardella. A simple MAX-CUT algorithm for planar graphs. Technical Report , 16 p., 2008
15. T.A. Davis. Direct Methods for Sparse Linear Systems, SIAM, Philadelphia, Sept. 2006.

Определение коллизий аппроксимирующих капсул и прямоугольных параллелепипедов

М.В. Михайлюк[1], **П.Ю. Тимохин**[2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail's : [1]*mix@niisi.ras.ru*, [2]*webpismo@yahoo.de*

Аннотация: Определение коллизий (столкновений) трехмерных виртуальных динамических объектов в системах компьютерного моделирования является важной задачей вычислительной динамики. В настоящей работе предлагается алгоритм определения коллизий аппроксимирующих капсул с аппроксимирующими прямоугольными параллелепипедами.

Ключевые слова: определение коллизий, аппроксимирующие контейнеры, моделирование динамики.

Введение

Одной из важных задач при моделировании трехмерных виртуальных динамических объектов (тел) является определение их столкновений (коллизий) [1]. Определить коллизии объектов произвольных форм достаточно трудно [2]. Поэтому один из распространенных методов определения коллизий основывается на том, что каждый виртуальный динамический объект окружается (аппроксимируется) одним или несколькими геометрическими контейнерами (прямоугольными параллелепипедами, сферами, капсулами и т.д.) [3]. В этом случае определение коллизии двух тел сводится к определению коллизий их аппроксимирующих контейнеров.

Алгоритмы определения коллизий условно можно разделить на два типа: априорные и апостериорные [4]. Априорные алгоритмы предсказывают столкновения тел до их фактического пересечения. Апостериорные алгоритмы определяют коллизии тел уже после их пересечения. В общем случае априорные алгоритмы определяют параметры коллизии с большей точностью. Но они требуют бóльшего объема исходных данных (положения и ориентации объектов, а также их физические характеристики - скорости, силы, моменты и т.д.). Поэтому эти алгоритмы имеют бóльшую вычислительную сложность и не позволяют моделировать динамику большого числа виртуальных объектов в режиме реального времени (не менее 25 шагов моделирования в секунду). Апостериорные алгоритмы определения коллизий используют в качестве входных данных лишь положения и ориентации объектов.

В настоящей работе предлагаются алгоритмы определения коллизий капсул с прямоугольными параллелепипедами (боксами). При этом мы предполагаем, что скорости объектов невелики, так, что за один шаг моделирования они не могут сильно проникнуть друг в друга. Мы будем считать, что глубина проникновения не превышает радиуса капсулы. Коллизии капсул между собой и со сферами рассмотрены в [8], а сфер и боксов между собой - в [5-6].

1. Постановка задачи

В этой работе мы будем использовать ортонормированные правосторонние системы координат. Каждый аппроксимирующий контейнер имеет матрицу перехода из его локальной системы координат (л.с.к.) в мировую (м.с.к.). Все вычисления будут производиться в мировой системе координат.

Начало O_1 локальной системы координат капсулы находится в нижней точке одной из ее полусфер, ось $\vec{z}$ направлена вдоль оси капсулы (см. рис. 1). Центры полусфер капсулы будем

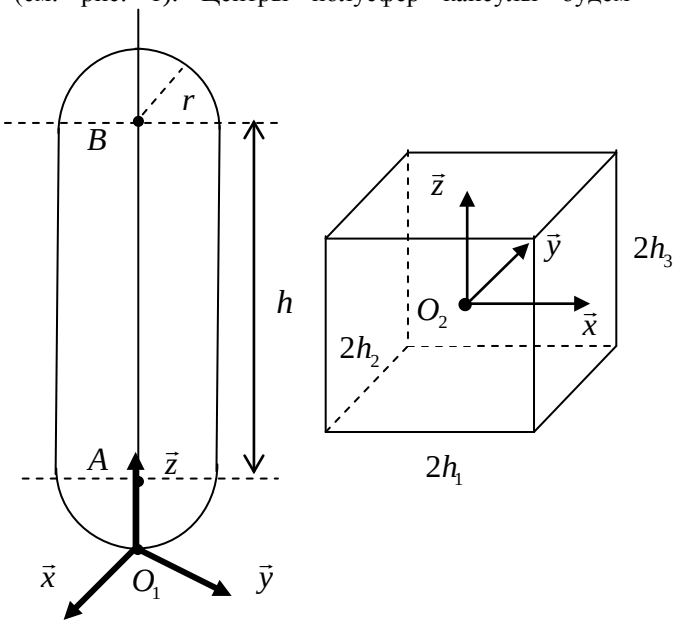


Рис. 1. Капсула $C(O_1,h,r)$ и бокс $E(O_2,h_1,h_2,h_3)$

обозначать через A и B, их радиусы - через r, расстояние между ними - h, а саму капсулу - через $C(O_1,h,r)$. Отрезок AB в дальнейшем будем называть осью капсулы. Начало O_2 локальной системы координат прямоугольного параллелепипеда (бокса) находится в его центре, оси координат $\vec{x},\vec{y},\vec{z}$ направлены параллельно ребрам, а по i-му

направлению ($i \in \{\vec{x}, \vec{y}, \vec{z}\}$) координаты вершин бокса равны $-h_i$ и h_i . Такой бокс будем обозначать $E(O_2, h_1, h_2, h_3)$.

Под определением коллизии пары аппроксимирующих контейнеров понимается определение факта их пересечения и, в случае его наличия, вычисление точки коллизии, нормали расталкивания и глубины коллизии. Точкой коллизии Q будем считать точку в м.с.к. на поверхности бокса, наиболее близкую к оси капсулы. Определим нормаль коллизии $\vec{N}$ как единичный вектор, направленный вдоль прямой, проходящей через точку контакта, и перпендикулярной оси капсулы или проходящей через ее вершину. Нормаль направлена внутрь капсулы, то есть указывает направление для ее выталкивания. Глубина коллизии d – это расстояние, на которое проникла точка коллизии внутрь капсулы вдоль нормали.

2. Определение коллизии капсулы и прямоугольного параллелепипеда

Нашей задачей будет поиск точек $L \in AB$ и $F \in E(O_2, h_1, h_2, h_3)$ таких, что расстояние между ними равно минимальному расстоянию между осью капсулы и поверхностью бокса. Запишем расстояние от поверхности бокса до оси капсулы в параметрическом виде:

$$| D(t) | = | P(t) - F(t) |,$$

где $P(t) = A + t(B - A)$ – точка на оси капсулы, а $F(t)$ – ближайшая к $P(t)$ точка на поверхности бокса. Это расстояние будет минимальным тогда, когда для производной квадрата расстояния $\left|D(\mathrm{x}(t), y(t), z(t))\right|^2$ будет выполнено

$$\frac{d\,|\,D(\mathrm{t})\,|^2}{dt} = \frac{\partial |D|^2}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial |D|^2}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial |D|^2}{\partial z} \cdot \frac{dz}{dt} = 0 . \qquad (1)$$

Вычислим значение этой производной для произвольной точки $P(t)$. Для этого разобьем пространство вне бокса на области: примыкающую к грани бокса, к ребру бокса или к его вершине. Рассмотрим три случая расположения точки $P(x, y, z)$, где (x, y, z) - ее координаты в системе координат бокса.

а) Точка $P(x, y, z)$ а расположена в области, примыкающей к грани бокса. Пусть, например, это будет область $x \ge h_x,\ -h_y \le y \le h_y,\ -h_z \le z \le h_z$ (см. рис. 2).

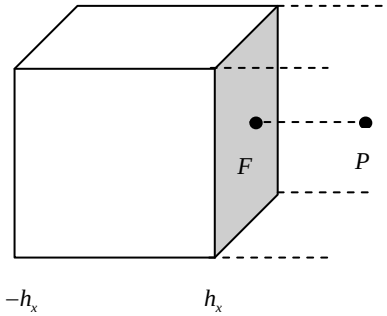


Рис. 2. Точка P находится в области, примыкающей к грани бокса

Тогда ближайшей к P точкой на кубе будет точка $F(h_x, y, z)$ (проекция P на грань куба) и

$$| D(P,F) |^2 = | P - F |^2 = (x - h_x)^2 ,$$

$$\frac{\partial |D|^2}{\partial x} = 2(x - h_x),\ \frac{\partial |D|^2}{\partial y} = 0,\ \frac{\partial |D|^2}{\partial z} = 0 .$$

Для остальных граней будет аналогично.

б) Точка $P(x, y, z)$ расположена в области, примыкающей к ребру бокса. Пусть, например, это будет область $x \ge h_x,\ y \le -h_y,\ -h_z \le z \le h_z$ (см. рис. 3).

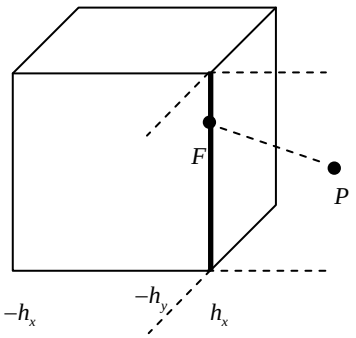


Рис. 3. Точка P находится в области, примыкающей к ребру бокса

Ближайшей к P точкой на кубе будет точка $F(h_x, -h_y, z)$ (проекция P на ребро куба) и

$$| D(P,F) |^2 = | P - F |^2 = (x - h_x)^2 + (\mathrm{y} - (-h_y))^2 ,$$

$$\frac{\partial |D|^2}{\partial x} = 2(x - h_x),\ \frac{\partial |D|^2}{\partial y} = 2(y - (-h_y)),\ \frac{\partial |D|^2}{\partial z} = 0 .$$

Для остальных ребер будет аналогично.

в) Точка $P(x, y, z)$ расположена в области, примыкающей к вершине бокса. Пусть, например, это будет область $x \ge h_x,\ y \le -h_y,\ z \ge h_z$ (см. рис. 4).

Ближайшей к P точкой на кубе будет его вершина $F(h_x, -h_y, h_z)$ и

$$| D(P,F) |^2 = | P - F |^2 = (x - h_x)^2 + (y - (-h_y))^2 + (z - h_z)^2 ,$$

$$\frac{\partial |D|^2}{\partial x} = 2(x - h_x),\ \frac{\partial |D|^2}{\partial y} = 2(y - (-h_x)),\ \frac{\partial |D|^2}{\partial z} = 2(z - h_z) .$$

Для остальных вершин будет аналогично. Заметим, что во всех случаях производная по i-й координате (где $i \in \{\vec{x}, \vec{y}, \vec{z}\}$) равна $2(i - a)$, где a - координата ближайшей к P грани, ребра или вершины.

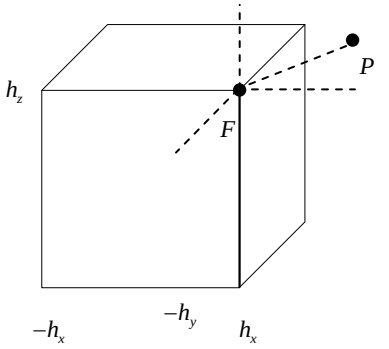


Рис. 4. Точка P находится в области, примыкающей к вершине бокса

Обозначим вектор $B - A$ оси капсулы через $\vec{v}$. Если P лежит на оси капсулы, т.е. $P = A + t\vec{v}$, то

$$\frac{dx}{dt} = v_x,\ \frac{dy}{dt} = v_y,\ \frac{dz}{dt} = v_z,$$

или, в общем виде для i-й координаты $\frac{di}{dt} = v_i$.

Замечание. Можно показать, что в общем случае производная (1) монотонно возрастает, принимая сначала отрицательные значения, затем нулевое и затем положительные значения.

Описание алгоритма. Обозначим через M_1 и M_2 матрицы перехода из локальных систем координат капсулы и бокса в мировую систему координат. Распишем алгоритм определения коллизии капсулы и бокса по пунктам.

1). Вычислим координаты точек A и B в м.с.к., а в л.с.к. бокса вычислим вектор $\vec{s}$, соединяющий центр бокса с точкой A капсулы и вектор $\vec{v} = \overrightarrow{AB}$. Имеем:

$A = M_1 \cdot (0,0,r) = r\vec{M}_{1,3}$;

$B = M_1 \cdot (0,0,h+r) = (h+r)\vec{M}_{1,3}$;

$\vec{s} = M_2^{-1} \cdot (A - O_2)$;

$\vec{v} = M_2^{-1} \cdot (B - A)$.

Здесь $\vec{M}_{1,3}$ - третий столбец матрицы M_1.

2). В силу симметрии можно отразить капсулу относительно осей координат бокса так, чтобы все компоненты вектора $\vec{v}$ стали неотрицательными

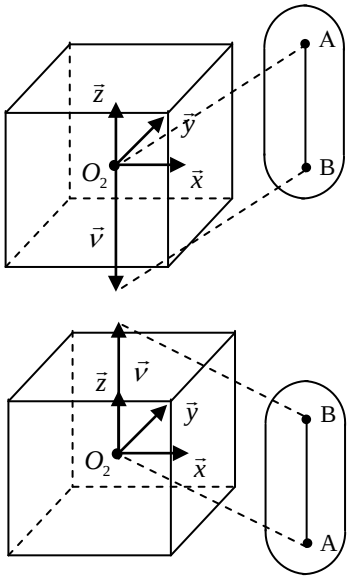


Рис. 5. Отражение вектора $\vec{v}$ относительно оси $\vec{z}$ бокса

(при этом вектор $\vec{s}$ также изменится). В массиве *sign*[3] будем записывать -1, если соответствующая компонента меняет знак и 1 в противном случае. Пример отражения относительно оси $\vec{z}$ бокса приведен на рис. 5. Алгоритм отражения будет иметь вид:

Цикл по i от 0 до 2

Если $v_i < 0$, то

$v_i = -v_i$;

$s_i = -s_i$;

$sign_i = -1$;

иначе $sign_i = 1$;

Конец цикла.

3). Для каждой i-ой оси бокса ($i \in \{x, y, z\}$) вычисляем значение t_i, соответствующее первому

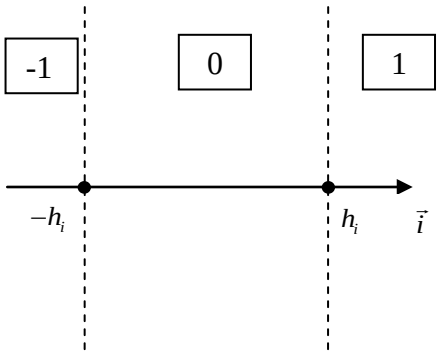


Рис. 6. Области проекции бокса на ось $\vec{i}$

пересечению оси капсулы $\vec{v}$ или ее продолжения с плоскостью грани бокса, перпендикулярной этой i-ой оси. Для этого каждую i-ую ось системы координат бокса разделим на три области: левее проекции бокса на эту ось, саму проекцию бокса и правее проекции бокса. Пронумеруем эти области соответственно числами -1, 0 и 1 (см. рис. 6) Для i-ой оси бокса вычислим код области, в которой лежит координата s_i (i-я координата точки A) и запомним этот код в элементе r_i массива r. Если проекция точки A на i-ую ось лежит в области с кодом (-1), то координата пересечения оси капсулы с плоскостью грани бокса будет равна $s_i + v_i \cdot t_i = -h_i$. Отсюда $t_i = (-h_i - s_i) / v_i$. Для 0-го и 1-го регионов эта координата будет равна $s_i + v_i \cdot t_i = h_i$, а $t_i = (h_i - s_i) / v_i$. Для тех осей, для которых $v_i = 0$ (т.е. для осей, перпендикулярных оси AB капсулы) код региона равен 0, а $t_i = 2$ (такое значение t_i далее нигде не учитывается). В результате получаем следующий алгоритм:

Цикл по i от 0 до 2

Если $v_i > \varepsilon$, то

Если $s_i < -h_i$, то

$r_i = -1$;

$t_i = (-h_i - s_i) / v_i$;

иначе

$r_i = (s_i > h_i)$;

$t_i = (h_i - s_i) / v_i$;

Конец если.

иначе

$r_i = 0$; $t_i = 2$.

Конец если.

Конец цикла.

4). Для каждой i-ой оси бокса вычислим значение $D_i = \frac{\partial |D|^2}{\partial i}$ и $D_i' = \frac{di}{dt}$ в точке $t = 0$ (соответствующей точке A) и значение общей производной $D'(t)$ в этой точке. Так как в дальнейшем нас будет интересовать только знак этой производной, то коэффициент 2 мы будем отбрасывать. Если точка A лежит в области с кодом (-1), то

$$D_i(0) = 2(s_i - (-h_i)) = 2(s_i - (s_i + t_i v_i)) = -2t_i v_i .$$

Тогда $D_i(0)D_i'(0) = -t_i v_i^2$.

Для 0-го региона проекция точки на i-ю ось находится внутри проекции бокса, поэтому $D_i = 0$ и $D[i]D'[i] = 0$. Для 1-го региона аналогично региону с кодом (-1) имеем:

$$D_i(0) = 2(s_i - h_i) = 2(s_i - (s_i + t_i v_i)) = -2 t_i v_i ;$$

$$D_i(0)D_i'(0) = -t_i v_i^2 .$$

Если общая производная больше 0, то переходим на пункт 6, так как в силу вышеприведенного Замечания при возрастании t производная будет только возрастать и, значит, ее минимум был в точке $t = 0$. Получаем следующий алгоритм:

$t = 0$.
$dd2dt = 0$. // общая производная
Цикл по i от 0 до 2
$dd2dt = dd2dt - (r_i \ ? \ t_i \cdot v_i^2 : 0)$
Конец цикла
Если $dd2dt \ge 0$, то перейти к п. 6.

5). Перебираем в порядке возрастания все значения t_i, которые соответствуют пересечению с гранями бокса проекции отрезка AB на i-ю ось. Если при переходе на следующий t_i производная меняет знак, то это означает, что между предыдущим и текущим t_i производная была равна 0. Тогда вычисляем соответствующее значение t и переходим на пункт 6. Обозначим через t_n минимальное значение среди t_i, $i \in \{x, y, z\}$. Аналогично п. 4, если точка $P(t_n)$ лежит в области с кодом (-1), то

$$D_i(t_n) = 2(s_i + v_i t_n - (-h_i)) = 2(s_i + v_i t_n - (s_i + v_i t_i)) =$$
$$= -2(t_i - t_n) \cdot v_i \quad \text{и}$$
$$D_i(t_n)D_i'(t_n) = -(t_i - t_n) \cdot v_i^2 .$$

Если точка $P(t_n)$ лежит в области с кодом 0, то ее проекция на i-ю ось находится внутри проекции бокса, поэтому $D_i(t_n) = 0$ и $D_i(t_n)D_i'(t_n) = 0$. Для 1-го региона аналогично региону с кодом (-1) имеем:

$$D_i(t_n) = 2(s_i + v_i t_n - h_i) = 2(s_i + v_i t_n - (s_i + v_i t_i)) =$$
$$= -2(t_i - t_n) \cdot v_i ;$$
$$D_i(t_n)D_i'(t_n) = -(t_i - t_n) \cdot v_i^2 .$$

Так как производная линейно зависит от t, то при смене ее знака значение t, при котором она равна нулю, легко вычисляется.

Цикл
$t = 0$; $t_n = 1$;
// вычисляем t_n - следующее значение t,
// отсекаемое гранью бокса
Цикл по i от 0 до 2
Если $t_i > t$ и $t_i < 1$ и $t_i < t_n$, то
$t_n = t_i$
Конец если
Конец цикла

// вычисляем значение полной производной
// в точке t_n
$next\_dd2dt = 0$;
Цикл по i от 0 до 2
$next\_dd2dt = next\_dd2dt -$
$-(r_i \ ? \ v_i^2 \cdot (t_i - t_n) : 0)$
Конец цикла
// если новая производная неотрицательна, то
// вычисляем t, в котором она равна нулю
Если $next\_dd2dt \ge 0$,то
$m = \frac{next\_dd2dt - dd2dt}{t_n - t}$
$t = t - dd2dt/m$
Перейти к п. 6.
Конец если
// переходим к следующему значению t_n
Цикл по i от 0 до 2
Если $t_i = t_n$, то
$t_i = (h_i - s_i) / v_i$
$r_i = r_i + 1$
Конец если
$t = t_n$
$dd2dt = next\_dd2dt$
пока $t < 1$
// если вышли из цикла по условию $t \ge 1$
$t = 1$

6). Вычисляем в м.с.к. для полученного итогового значения t ближайшую к точке $L = P(t)$ оси капсулы точку F на поверхности бокса.

// вычисляем точку L на оси капсулы
$L = A + t \cdot \vec{v}$;
// вычисляем точку F на боксе
Цикл по i от 0 до 2
$tmp_i = sign_i \cdot (s_i + t_i \cdot v_i)$;
Если $tmp_i < -h_i$, то $tmp_i = -h_i$;
иначе если $tmp_i > h_i$, то $tmp_i = h_i$;
Конец цикла.
$\overrightarrow{tmp} = M_2 \cdot \overrightarrow{tmp}$;
$F = O_2 + \overrightarrow{tmp}$.

7). В силу сделанного ранее предположения о том, что за один шаг моделирования капсула не может проникнуть внутрь бокса более чем на величину радиуса, точка L будет лежать вне бокса. Поэтому результирующие характеристики коллизии (точка, нормаль и глубина проникновения) можно вычислить непосредственно. Направление вектора нормали $\vec{N}$ будет совпадать с направлением вектора $\overrightarrow{FL}$, глубина $d = r - |FL|$, а точка контакта $Q = F$.

Заключение

Описанные алгоритмы определения коллизий капсул с прямоугольными параллелепипедами (боксами) позволяют рассчитывать информацию о коллизиях с небольшими вычислительными затратами. Совместно с описанными в [7] алгоритмами разрешения коллизий, учитывающими контакт, удар и трение, это позволяет моделировать динамику большого (несколько тысяч) числа виртуальных объектов в режиме реального времени.

Рассмотренные алгоритмы были реализованы в рамках разработанной в ФГУ ФНЦ НИИСИ РАН системы моделирования динамики виртуальных объектов. Апробация алгоритмов в рамках этой системы динамики показала реалистичность поведения виртуальных объектов и приемлемое время расчетов.

Работа выполнена при поддержке РФФИ, грант № 16-37-00218.

Collision detection for bounding capsules and boxes

M.V. Mikhaylyuk, P.Yu. Timokhin

Abstract. Collision detection of 3D virtual dynamic objects in computer simulation systems is a quite important task of computational dynamics. In this paper, we propose an algorithm for collision detection between capsules and boxes.

Keywords: collision detection, bounding volumes, dynamics simulation

Литература

1. M. Moore, J. Wilhelms. Collision Detection and Response for Computer Animation. Computer Graphics, 1998, Vol. 22, no. 4, pp. 289–298. Available from: http://www.cs.princeton.edu/courses/archive/spring01/cs598b/papers/moore88.pdf

2. C. Lin Ming, S. Gottschalk. Collision detection between geometric models: a survey. Proc. of IMA conference on mathematics of surfaces. 1998, Vol. 1, pp. 602–608. Available from: https://users.soe.ucsc.edu/~pang/161/w06/notes/ cms98.pdf .

3. Bounding volume. Wikipedia. The Free Encyclopedia. Available from: https://en.wikipedia.org/wiki/Bounding_volume .

4. C. Ericson. Real-Time Collision Detection. CRC Press, 2004, 594 p. Available from: https://books.google.ru/books?id=WGpL6Sk9qNAC .

5. М.В. Михайлюк, А.М. Трушин. Расчет коллизий прямоугольных параллелепипедов в задачах динамики. Труды НИИСИ РАН. Том 2, №2. Обработка изображений, моделирование и визуализация: теоретические и прикладные аспекты. М.: НИИСИ РАН, 2012. С. 51–59.

6. А.М. Трушин. Определение коллизий аппроксимирующих сфер и прямоугольных параллелепипедов в системах трехмерного моделирования. Программные продукты и системы – 2015. – №4. – С. 105-109.

7. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов. Труды НИИСИ РАН. Том 4, №2. Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты. М.: НИИСИ РАН, 2014. С. 95–105.

8. А.М. Трушин, М.В. Михайлюк, Определение коллизий аппроксимирующих капсул и сфер трехмерных виртуальных динамических объектов. Труды НИИСИ РАН. Том 6, № 2. Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты – М.: НИИСИ РАН, 2016. – С. 42-45.

Метод сжатия разрядности карт высот на основе критерия визуальной значимости

П.Ю. Тимохин [1], **М.В. Михайлюк** [2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail's : [1] *webpismo@yahoo.de,* [2] *mix@niisi.ras.ru,*

Аннотация. В статье предлагается критерий визуальной значимости высот виртуального рельефа Земли, учитывающий направление, высоту, угол обзора и разрешение средства наблюдения, а также кривизну земной поверхности. Также в статье описывается новый метод сжатия разрядности карт высот, основанный на разработанном критерии, который обеспечивает представление высотных данных на синтезируемых изображениях земной поверхности без заметной потери качества. Предлагаемое решение позволяет эффективно уменьшать объем карт высот, а также их размеры (при упаковке в текстуры большей разрядности). Разработанный критерий и метод были успешно апробированы в подсистеме визуализации виртуальной модели Земли в имитационно-тренажерном комплексе, разработанном в ФГУ ФНЦ НИИСИ РАН.

Ключевые слова: карта высот, разрядность, сжатие, виртуальный рельеф, визуализация

Введение

Одной из важных задач многих современных космических видеотренажеров является моделирование и визуализация в реальном времени виртуального рельефа Земли на основе детализированных *карт высот* [1]. Карта высот – это 2D-текстура, в каждом текселе которой записана высота соответствующей точки рельефа относительно поверхности уровня. В данной работе рассматриваются целочисленные карты высот разрядностью 8, 16, 24 и 32 бит.

Для моделирования земной поверхности, наблюдаемой из космоса невооруженным глазом (через иллюминатор космического корабля), детализация текстурных карт должна быть не менее 150 м/пиксел (размера порядка 10^5x10^5 текселов), и на порядок выше для средств наблюдения с высокой кратностью увеличения [2]. Размеры и объем таких текстур намного превышают ограничения современных видеокарт (максимальный размер - 16Кх16К текселов), что не позволяет загружать их целиком в видеопамять и обрабатывать аппаратно. Эффективным путем является подкачка в видеопамять только тех кусков текстуры, которые необходимы для визуализации. В этой связи необходимы эффективные методы и алгоритмы сжатия карт высот, обеспечивающие выполнение подкачки в реальном времени.

В данной области многие работы [3] исследуют различные способы сжатия размеров карт высот. Однако при таком подходе не учитывается изменение различимости на экране (*визуальной значимости*) перепадов высот в зависимости от направления, высоты, угла обзора и разрешения средства наблюдения (виртуальной камеры), а также кривизны земной поверхности. В результате этого во многих сжатых текстурах перепады высот хранятся с избыточной точностью, которую наблюдатель на экране фактически не увидит.

В данной работе предлагается критерий визуальной значимости высот рельефа, учитывающий описанный выше фактор, и разработанный на его основе метод сжатия разрядности карт высот без заметной потери качества представления высотных данных на синтезируемых изображениях земной поверхности.

1. Критерий визуальной значимости высот рельефа

Пусть в точке P расположена виртуальная камера с вертикальным углом обзора α градусов, экран размера D пикселов, и некоторая точка A виртуального рельефа Земли с высотой h относительно сферы радиуса R (рис. 1).

Виртуальная камера расположена на высоте H от сферы ($H > h$) и направлена в основание B высоты точки A под углом β к надиру.

Проведем лучи из точек A и B через P до пересечения с картинной плоскостью камеры. В полученном прямоугольном ΔPEF обозначим через p сторону EF. Назовем величину p размером высоты h на экране, в пикселах.

Введем следующий ***Критерий визуальной значимости высоты рельефа:***

Высота h рельефа является визуально значимой, если при заданных значениях α, D, R и H ее наибольший размер p_{max} не меньше порогового значения p_{thres}, установленного пользователем.

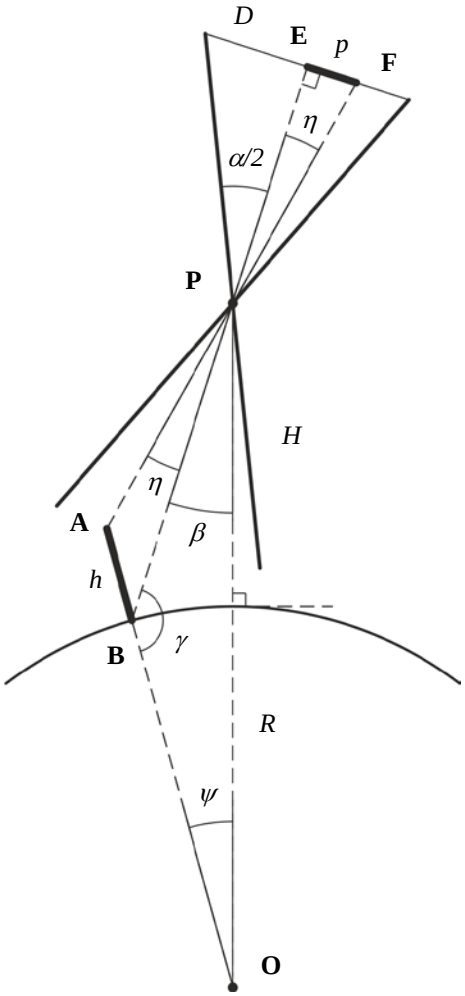


Рис. 1. Схема отображения высоты h рельефа в виртуальной камере

Как видно из рисунка 1, значение величины p зависит от угла β, который принимает значения из отрезка $[0, \beta_{hor}]$, где $\beta_{hor} = \arcsin\dfrac{R}{R+H}$ - угол между надиром и направлением взгляда камеры в горизонт (касательной к сфере) [4]. Тогда наибольший размер p_{max} можно найти как наибольшее значение функции $p(\beta)$ на отрезке $[0, \beta_{hor}]$. Рассмотрим это более подробно.

Нахождение функции $p(\beta)$. Величину p можно выразить из ΔPEF (см. рис. 1) как

$$p = f \cdot \operatorname{tg}\eta, \qquad (1)$$

где $f = \dfrac{D}{2\operatorname{tg}(\alpha/2)}$ - длина стороны PE в ΔPEF.

Отметим, что в случае, когда $\beta = 0$ или $h = 0$ ($\eta = 0$), т.е. когда высота h вырождается в точку на картинной плоскости, мы полагаем $p = 1$, поэтому здесь и далее рассматривается случай, когда $\beta \in (0, \beta_{hor}]$ и $h > 0$.

В формуле (1) величину $\operatorname{tg}\eta$ найдем из ΔPBO и ΔPBA. Согласно теореме синусов для данных треугольников справедливы соотношения

$$\frac{PB}{\sin\psi} = \frac{R}{\sin\beta}, \qquad \frac{PB}{\sin(\gamma-\eta)} = \frac{h}{\sin\eta}.$$

Выражая из них величины PB и приравнивая их друг к другу, получим:

$$\frac{R}{\sin\beta}\sin\psi = \frac{h}{\sin\eta}\sin(\gamma-\eta)$$

или

$$\frac{R}{\sin\beta}\sin(\beta+\gamma) = \frac{h}{\sin\eta}\sin(\gamma-\eta).$$

В полученном равенстве разложим $\sin(\beta+\gamma)$ и $\sin(\gamma-\eta)$ по формулам сложения и вычитания аргументов:

$$R(\cos\gamma + \operatorname{ctg}\beta\cdot\sin\gamma) = h(\sin\gamma\cdot\operatorname{ctg}\eta - \cos\gamma).$$

Разделим обе части полученного выражения на $\cos\gamma$ и, выполнив несложные тригонометрические преобразования, получим формулу для $\operatorname{tg}\eta$:

$$\operatorname{tg}\eta = \frac{h}{R\cdot\operatorname{ctg}\beta + (R+h)\cdot\operatorname{ctg}\gamma}. \qquad (2)$$

В формуле (2) неизвестен $\operatorname{ctg}\gamma$. Из рисунка 1 следует, что угол γ принадлежит II-ой координатной четверти, т.е. $\cos\gamma < 0$. Тогда для $\operatorname{ctg}\gamma$ справедливо следующее выражение:

$$\operatorname{ctg}\gamma = \frac{\cos\gamma}{\sin\gamma} = \frac{-\sqrt{1-\sin^2\gamma}}{\sin\gamma} = -\sqrt{\frac{1}{\sin^2\gamma}-1}. \qquad (3)$$

Величину $\sin\gamma$ найдем из ΔPBO согласно теореме синусов:

$$\sin\gamma = \frac{R+H}{R}\sin\beta. \qquad (4)$$

Подставив выражение (4) в формулу (3), получим:

$$\operatorname{ctg}\gamma = -\sqrt{\frac{R^2}{(R+H)^2\sin^2\beta}-1} = -\sqrt{\frac{R^2}{(R+H)^2}(\operatorname{ctg}^2\beta+1)-1}. \qquad (5)$$

Обозначим через G величину $\frac{1}{R+H}$, а через J - величину $\frac{(R+H)^2}{R^2}-1$. Подставив в формулу (1) выражения (2) и (5), получим выражение для функции $p(\beta)$:

$$p=\frac{f}{R}\cdot\frac{h}{\operatorname{ctg}\beta-G(R+h)\sqrt{\operatorname{ctg}^2\beta-J}}. \qquad (6)$$

Нахождение величины p_{max}. Чтобы найти наибольшее значение p_{max} функции $p(\beta)$, найдем все критические точки функции $p(\beta)$ в интервале $(0, \beta_{hor})$, вычислим значения функции в этих точках, а также при $\beta = \beta_{hor}$, и выберем из них наибольшее.

Для этого вычислим производную $p'(\beta)$. В формуле (6) обозначим через K величину $\frac{f\cdot h}{R}$, через L - величину $G(R+h)$, а через $t(\beta)$ - выражение $\operatorname{ctg}\beta-L\sqrt{\operatorname{ctg}^2\beta-J}$ (величина $t(\beta)>0$, так как $J>0$ и $L<1$). Тогда $p'(\beta)$ можно записать как

$$p'(\beta)=K\left(\frac{1}{t(\beta)}\right)'=-K\frac{t'(\beta)}{t^2(\beta)}=$$

$$=-\frac{K\left(-\frac{1}{\sin^2\beta}+L\frac{\operatorname{ctg}\beta}{\sin^2\beta\sqrt{\operatorname{ctg}^2\beta-J}}\right)}{t^2(\beta)}= \qquad (7)$$

$$=\frac{K\left(1-L\frac{\operatorname{ctg}\beta}{\sqrt{\operatorname{ctg}^2\beta-J}}\right)}{\sin^2\beta\cdot t^2(\beta)}.$$

Приравняв производную (7) к 0, получим уравнение

$$\sqrt{\operatorname{ctg}^2\beta-J}=L\operatorname{ctg}\beta.$$

Возведем в квадрат обе его части и, приведя подобные члены, получим:

$$\operatorname{ctg}^2\beta=\frac{J}{1-L^2}. \qquad (8)$$

Из рисунка 1 следует, что угол β принадлежит I-ой координатной четверти, т.е. $\operatorname{ctg}\beta>0$, поэтому квадратное уравнение (8) имеет один корень

$$\operatorname{ctg}\beta=\sqrt{\frac{J}{1-L^2}},$$

из которого получаем выражение для критической точки $\beta_{crit,p}$ функции $p(\beta)$:

$$\beta_{crit,p}=\operatorname{arcctg}\sqrt{\frac{J}{1-L^2}}. \qquad (9)$$

Критические точки, в которых производная (7) не существует, не входят в интервал $(0, \beta_{hor})$, поэтому мы их отбрасываем.

Вычислим значения $p(\beta_{crit,p})$ и $p(\beta_{hor})$ и сравним их между собой. Для этого подставим выражение (9) в формулу (6) и получим:

$$p(\beta_{crit,p})=K\frac{1}{\sqrt{\frac{J}{1-L^2}}-L\sqrt{\frac{J}{1-L^2}-J}}=$$

$$=K\frac{1}{\sqrt{J\cdot(1-L^2)}}$$

Запишем выражение для β_{hor} как

$$\beta_{hor}=\arcsin\frac{R}{R+H}=\operatorname{arcctg}\sqrt{J}. \qquad (10)$$

Подставив выражение (10) в формулу (6), получим:

$$p(\beta_{hor})=K\frac{1}{\sqrt{J}}. \qquad (11)$$

Так как $H > h$ (из условия задачи), то $L\in(0,1)$, а $p(\beta_{crit,p})>p(\beta_{hor})$. Тогда получим следующее выражение для величины p_{max}:

$$p_{max}=p(\beta_{crit,p})=K\frac{1}{\sqrt{J\cdot(1-L^2)}}. \qquad (12)$$

На рисунке 2 показан пример графика функции $p(\beta)$, построенный по формуле (6), для $h = 8848$ м (г. Эверест) при $D = 2048$ пикселов, $\alpha = 64°$ (телекамера), $R = 6371$ км, $H = 400$ км. Пунктиром на графике отмечены значения $\beta_{crit,p} = 42{,}95°$ и $p_{max} = 19$ пикселов, вычисленные по формулам (9) и (12).

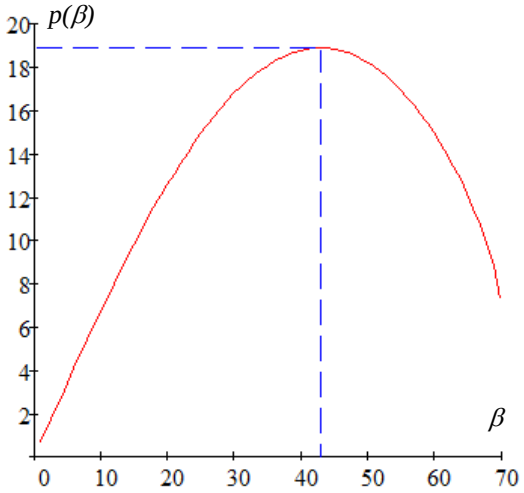


Рис. 2. График функции $p(\beta)$

2. Эффективная разрядность карт высот

Обратимся снова к рисунку 1. Обозначим через h_{thres} некоторую высоту h, которая имеет размер $p = p_{thres}$ пикселов на экране. Как и в разделе 1, зафиксируем значения параметров α, D, R и H, тогда величина h_{thres} будет

зависеть только от угла β. Обозначим через $h_{thres,min}$ наименьшее значение $h_{thres}(\beta)$ на отрезке $[0, \beta_{hor}]$. Введем следующее определение:

Эффективная разрядность d_{eff} карты высот, т.е. количество двоичных разрядов, необходимых для хранения наибольшей высоты h_{max} рельефа с точностью до $h_{thres,min}$ равно

$$d_{eff} = \left\lceil \log_2 \left\lceil \frac{h_{\max}}{h_{\text{thres,min}}} \right\rceil \right\rceil. \qquad (13)$$

Чтобы вычислить значение $h_{thres,min}$, найдем функцию $h_{thres}(\beta)$ и определим ее наименьшее значение на отрезке $[0, \beta_{hor}]$. Рассмотрим это более подробно.

Нахождение функции $h_{thres}(\beta)$. Функцию $h_{thres}(\beta)$ найдем из формулы (6). Для этого выразим из (6) величину h и подставим в полученное выражение значение p_{thres}:

$$h_{thres} = \frac{\operatorname{ctg}\beta - GR\sqrt{\operatorname{ctg}^2\beta - J}}{Q + G\sqrt{\operatorname{ctg}^2\beta - J}} =$$

$$= \frac{\operatorname{ctg}\beta - GR\sqrt{\operatorname{ctg}^2\beta - J} + RQ - RQ}{Q + G\sqrt{\operatorname{ctg}^2\beta - J}} = \qquad (14)$$

$$= \frac{\operatorname{ctg}\beta + RQ}{Q + G\sqrt{\operatorname{ctg}^2\beta - J}} - R,$$

где $Q = \dfrac{f}{p_{thres} R}$.

Нахождение величины $h_{thres,min}$. Чтобы найти $h_{thres,min}$ найдем все критические точки функции $h_{thres}(\beta)$ в интервале $(0, \beta_{hor})$, вычислим значения функции в этих точках, а также при $\beta = \beta_{hor}$, и выберем из них наименьшее.

Для этого вычислим производную $h_{thres}'(\beta)$. Обозначим через $u(\beta)$ числитель дроби в формуле (14), а через $v(\beta)$ - ее знаменатель ($v(\beta) > 0$, т.к. $Q > 0$ и $\beta < \operatorname{arcctg}\sqrt{J}$). Тогда $h_{thres}'(\beta)$ можно записать как

$$h_{thres}'(\beta) = \left(\frac{u(\beta)}{v(\beta)}\right)' =$$

$$= \frac{G\dfrac{u(\beta)\cdot\operatorname{ctg}\beta}{\sin^2\beta\sqrt{\operatorname{ctg}^2\beta - J}} - \dfrac{v(\beta)}{\sin^2\beta}}{v^2(\beta)} = \qquad (15)$$

$$= \frac{G\dfrac{u(\beta)\cdot\operatorname{ctg}\beta}{\sqrt{\operatorname{ctg}^2\beta - J}} - v(\beta)}{v^2(\beta)\sin^2\beta}.$$

Приравняв производную (15) к нулю, получим уравнение:

$$\sqrt{\operatorname{ctg}^2\beta - J} = \frac{G\cdot\operatorname{ctg}\beta\cdot(\operatorname{ctg}\beta + RQ)}{Q + G\sqrt{\operatorname{ctg}^2\beta - J}}$$

или

$$\sqrt{\operatorname{ctg}^2\beta - J} = G\cdot\left(R\operatorname{ctg}\beta + \frac{J}{Q}\right). \qquad (16)$$

Возведя в квадрат обе части уравнения (16), приведя подобные члены и используя замену $G^2R^2 = \dfrac{1}{J+1}$ (из раздела 1), получим квадратное уравнение относительно $\operatorname{ctg}\beta$:

$$(QR\operatorname{ctg}\beta - 1)^2 = (1+J)(1+Q^2R^2). \qquad (17)$$

Уравнение (17) имеет только один корень (т.к. $\operatorname{ctg}\beta > 0$ и $J > 0$, см. раздел 1):

$$\operatorname{ctg}\beta = \frac{1 + \sqrt{(1+J)(1+Q^2R^2)}}{QR},$$

из которого получаем выражение для критической точки $\beta_{crit,h}$ функции $h_{thres}(\beta)$:

$$\beta_{crit,h} = \operatorname{arcctg}\frac{1 + \sqrt{(1+J)(1+Q^2R^2)}}{QR}. \qquad (18)$$

Критические точки, в которых производная (15) не существует, не входят в интервал $(0, \beta_{hor})$, поэтому мы их отбрасываем.

Вычислим значения $h_{thres}(\beta_{hor})$ и $h_{thres}(\beta_{crit,h})$ и сравним их между собой. Для этого подставим выражение (10) в формулу (14) и получим:

$$h_{thres}(\beta_{hor}) = \frac{\sqrt{J}}{Q}. \qquad (19)$$

Также подставим выражение (18) в формулу (14) и, упростив его, получим выражение

$$h_{thres}(\beta_{crit,h}) = \frac{1 + \sqrt{(1+J)(1+Q^2R^2)} + Q^2R^2}{Q^2R + G\left(\sqrt{1+J} + \sqrt{1+Q^2R^2}\right)} - R,$$

из которого, используя замену $J + 1 = \dfrac{1}{G^2R^2}$, получим:

$$h_{thres}(\beta_{crit,h}) = \frac{1 - G^2R^2}{G\left(GR + \sqrt{1+Q^2R^2}\right)} =$$

$$= \frac{JR}{1 + \sqrt{(1+J)(1+Q^2R^2)}}.$$

Сравним величины $h_{thres}(\beta_{hor})$ и $h_{thres}(\beta_{crit,h})$:

$$\frac{h_{thres}(\beta_{hor})}{h_{thres}(\beta_{crit,h})} = \frac{1 + \sqrt{(1+J)(1+Q^2R^2)}}{QR\sqrt{J}} = \frac{\operatorname{ctg}\beta_{crit,h}}{\operatorname{ctg}\beta_{hor}}.$$

Так как $\beta_{crit,h} < \beta_{hor}$, то $h_{thres}(\beta_{crit,h}) < h_{thres}(\beta_{hor})$, поэтому:

$$h_{thres,\min} = h_{thres}\left(\beta_{crit,h}\right) = \frac{JR}{1+\sqrt{\left(1+J\right)\left(1+Q^2R^2\right)}}. \quad (20)$$

На рисунке 3 приведен пример графика функции $h_{thres}(\beta)$, построенный по формуле (14), для $p_{thres} = 1$ пиксел при тех же D, α, R и H, что и график на рис. 2. Пунктиром на графике отмечены значения $\beta_{crit,h} = 43{,}24°$ и $h_{thres,min} = 473{,}49$ метров, вычисленные по формулам (18) и (20).

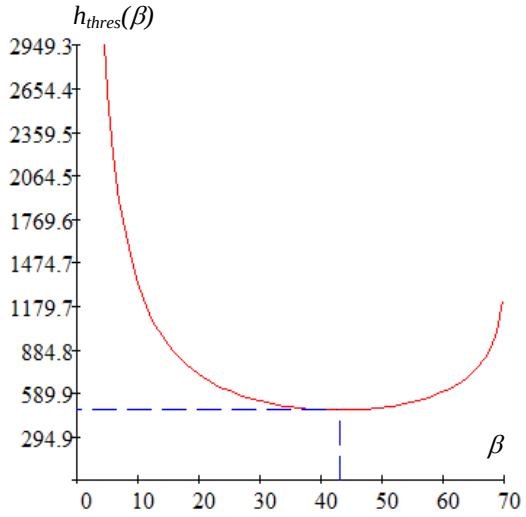


Рис. 3. График функции $h_{thres}(\beta)$

3. Результаты

С помощью полученных в данной работе формул (12), (13), и (20) был выполнен расчет величины p_{max} визуальной значимости ряда типовых высот рельефа Земли, а также расчет точности $h_{thres,min}$ и эффективной разрядности d_{eff} карт высот для видеотренажеров Международной космической станции (МКС).

В таблице 1 приведены результаты расчета p_{max} (в пикселах) для угла α обзора иллюминатора (103°), телекамеры с широким углом обзора (64°), телекамеры с узким углом обзора (16°), фотокамеры с 10-кратным зумом (3°). Расчеты были выполнены при R = 6371 км, H = 400 км, D = 2048 пикселов.

Таблица 1. Результаты расчета p_{max}

h, м \ α,°	103°	64°	16°	3°
500	1	2	5	26
1000	2	3	10	51
2000	3	5	19	101
4000	5	9	38	203
8848	10	19	84	451

Из таблицы 1 видно, что высоты рельефа до 1 км (~72% суши) при $\alpha = 103°$ и $p_{thres} = 2$ пиксела на экране практически не различимы, и их значения можно округлить одной константой, что существенно упрощает моделирование земной поверхности. Визуальная значимость наибольшей высоты Земли при увеличении угла обзора от 3° до 103° уменьшается в разы, что дает основания для уменьшения разрядности карты высот.

В таблицах 2 и 3 приведены значения величин $h_{thres,min}$ (в метрах) и d_{eff} (в битах), вычисленные по формулам (20) и (13) для ряда типовых размеров D экрана и углов α обзора, вычисленные при самом точном пороге p_{thres} = 1 пиксел.

Таблица 2. Результаты расчета $h_{thres,min}$

D, пикс. \ α,°	103°	64°	16°	3°
1024	1902	946	213	40
2048	952	473	107	20
4096	476	237	53	10

Таблица 3. Результаты расчета d_{eff}

D, пикс. \ α,°	103°	64°	16°	3°
1024	3	4	6	8
2048	4	5	7	9
4096	5	6	8	10

Заключение

В работе предложен критерий визуальной значимости высот рельефа, учитывающий направление, высоту, угол обзора и разрешение средства наблюдения, а также кривизну земной поверхности. На основе предложенного критерия был разработан метод сжатия разрядности карт высот, основанный на вычислении эффективной разрядности, который не дает заметной потери качества представления высотных данных на синтезируемых изображениях земной поверхности. Апробация разработанного решения в подсистеме визуализации виртуальной поверхности Земли показала его успешную применимость для решения задач, связанных с визуализацией рельефа в имитационно-тренажерных комплексах космического назначения. Исследование было выполнено при финансовой поддержке РФФИ в рамках проекта № 17-07-00243.

The method of height map bit depth compression based on visual significance criterion

P.Yu. Timokhin, M.V. Mikhaylyuk

Abstract. The paper proposes a criterion of visual significance of heights of the virtual Earth relief, which takes into account the direction, altitude, view angle and resolution of observation tool, as well as the curvature of the Earth's surface. Also the article describes a new method of height map bit depth compression based on developed criterion, which provides height data representation without noticeable loss of quality on the synthesized images of the Earth's surface. The proposed solution allows to effectively reduce height map volume, as well as it's dimensions (when packaged in the texture with larger bit depth). Developed criterion and method were successfully tested in virtual Earth visualization subsystem of training complex developed in SRISA RAS.

Keywords: height map, bit depth, compression, virtual relief, visualization

Литература

1. М.В. Михайлюк, П.Ю. Тимохин, А.В. Мальцев. Адаптивная тесселяция на GPU виртуального рельефа с помощью патчей-треугольников. «Труды 26-й Международной конференции GraphiCon2016», Нижний Новгород, Изд-во ИФТИ, ННГАСУ, 2016, 39 - 43.
2. П.Ю. Тимохин. Система визуализации текстурированных моделей планет для тренировок проведения космических экспериментов. // Программные продукты и системы. – 2015. – № 4. – С. 99-104.
3. M.Lasan. Height map compression techniques. // Master Thesis, Charles University, Prague, 2016.
4. М.Н. Бурдаев. О форме границ и размерах зон обзора поверхностей планет с космических аппаратов. // Пилотируемые полеты в космос. 2014. № 3. с. 71-75.

Моделирование разрыва шарниров виртуальных роботов

Е.В. Страшнов[1], **М.А. Торгашев**[2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»,
E-mail's: [1]strashnov_evg@mail.ru, [2]mtorg@mail.ru

Аннотация: Рассматривается задача моделирования поведения виртуального робота при возникновении внештатных ситуаций, связанных с разрушением шарнирных соединений под действием внешних нагрузок. С использованием метода последовательных импульсов допустимая нагрузка, которую может выдержать шарнир, описывается ограничениями, накладываемыми на накапливаемые в шарнире импульсы. При поломке шарниров виртуального робота предлагается обрабатывать аварийную ситуацию, при которой отключаются некоторые двигатели робота. Апробация предложенных в статье алгоритмов и методов была проведена в подсистеме динамики имитационно-тренажерного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН.

Ключевые слова: разрыв шарнира, внештатные ситуации, метод последовательных импульсов, имитационно-тренажерный комплекс.

Введение

Одним из направлений робототехники является использование роботов в экстремальных внешних условиях, не пригодных для деятельности человека (радиационное заражение, пожар и т.п.). Робот в таких условиях должен выполнять ряд технологических операций, таких как инспекция, погрузо-разгрузочные работы, манипуляционные работы и т.д. При этом управление роботом может осуществляться специально обученным оператором с помощью пульта управления. При таком режиме управления в процессе выполнения операций возможно возникновение различных внештатных ситуаций. Одним из возможных вариантов такой ситуации является возникновение больших сил в сочленении робота, способных привести к поломке шарнира или двигателя. Такие ситуации возникают при сильном ударе (например, падение робота с высоты), при попытке взять манипулятором слишком тяжелый объект и при некачественном управлении в отсутствии датчиков усилий для контроля сил и моментов. В реальных системах при подобных условиях срабатывает защита двигателей робота от перегрузки, что приводит к невозможности выполнения технологических операций. Для предотвращения возникновения подобных ситуаций операторы тренажера должны пройти квалифицированное обучение навыкам управления роботами, но при этом использование реальных роботов в процессе обучения является рискованным и затратным. В качестве альтернативного решения предлагается проводить процесс обучения на виртуальных моделях роботов в виртуальной среде. Поэтому для виртуальных роботов необходимо моделировать возникновение внештатных ситуаций, связанных с поломкой роботов.

Моделирование поломки шарниров активно используется в игровых и физических движках [1, 2]. Так, в игровом движке Unity [1], чтобы сделать поведение игровых объектов более реалистичным, для различных типов шарнира задаются параметры разрушения. Примером служит реализация Ragdoll-физики [3] (физики тряпичной куклы) в виде набора твердых тел, связанных друг с другом шарнирами. Разрыв шарнира происходит в таких игровых сценариях, когда персонаж подвергается взрыву от ракетной установки или на него на скорости наезжает автомобиль. Аналогичный подход используется и в физическом движке PhysX [2].

В данной статье предлагается моделировать разрыв шарниров виртуальных роботов в имитационно-тренажерных комплексах. Для моделирования динамики виртуальных роботов применяются методы моделирования шарнирно-связанных тел на основе последовательных импульсов [4, 5]. Здесь предлагается усовершенствовать данный метод для ограничения нагрузки, которой подвержен шарнир. При превышении допустимой нагрузки для робота предусмотрена аварийная ситуации, когда шарнир разрывается и происходит отключение некоторых двигателей с изменением их характеристик.

1. Моделирование разрыва шарнира

Разрыв шарнира возможен в двух ситуациях: когда превышена сила в шарнире или когда превышен момент в шарнире. Для моделирования разрыва шарнира предлагается использовать два параметра: максимальная сила разрыва $F_{\max}$ и максимальный момент разрыва $\tau_{\max}$, при превышении которых шарнир ломается. Тогда условия разрыва шарнира примут вид

$$\left|\overline{F}\right| > F_{\max},\ \left|\overline{\tau}\right| > \tau_{\max}, \qquad (1)$$

где $\overline{F}$ и $\overline{\tau}$ - сила и момент нагрузки на шарнир.

При моделировании шарнирно-связанных тел на основе последовательных импульсов [4] обрабатываются ограничения на положения и ориентации соединяемых тел. Например, осевой шарнир задается 3 ограничениями на положение тел и 2 ограничениями на ориентации. В методе последовательных импульсов для обеспечения ограничений на положения тел вычисляется импульс силы $\overline{p}_f$, а для обеспечения ограничений на ориентацию тел вычисляется импульс момента $\overline{p}_\tau$. Умножая (1) на шаг моделирования Δt запишем условия разрыва шарнира в виде

$$\left|\overline{p}_f\right| > p_{f,\max},\ \left|\overline{p}_\tau\right| > p_{\tau,\max}, \qquad (2)$$

где $p_{f,\max} = F_{\max}\Delta t$ - максимальный импульс силы разрыва шарнира, $p_{\tau,\max} = \tau_{\max}\Delta t$ - максимальный импульс момента разрыва шарнира, Δt - шаг моделирования.

С учетом разрыва шарнира алгоритм метода последовательных импульсов (приведенный в [4]) для системы тел, состоящей из N шарниров, примет следующий вид:

Для каждого j-го шарнира ($j = 1,...,N$) :
  $isBreak[j] = false$
На i-м шаге моделирования:
  Цикл по всем шарнирам $j = 1,...,N$
    Если $!isBreak[j]$, то
      - применяем накопленные импульсы $\overline{p}_f$, $\overline{p}_\tau$
      - вычисляем $p_{f,\max}$ и $p_{\tau,\max}$
  Конец цикла по шарнирам
  Цикл по итерациям метода последовательных импульсов
    Цикл по всем шарнирам $j = 1,...,N$
      Если $!isBreak[j]$, то
        - вычисляем импульсы $\Delta\overline{p}_f$, $\Delta\overline{p}_\tau$
        - накапливаем импульсы:
          $\overline{p}_f := \overline{p}_f + \Delta\overline{p}_f$
          $\overline{p}_\tau := \overline{p}_\tau + \Delta\overline{p}_\tau$
        - применяем импульсы $\Delta\overline{p}_f$, $\Delta\overline{p}_\tau$
    Конец цикла по шарнирам
  Конец цикла по итерациям
  Цикл по всем шарнирам $j = 1,...,N$
    Если $!isBreak[j]$, то
      Если выполнено одно из условий (2), то
        разрываем шарнир
        $isBreak[j] = true$
  Конец цикла по шарнирам

В данном алгоритме вычисление реакций связей в шарнире на каждом шаге моделирования динамики системы тел продолжается, пока флаг поломки j-го шарнира $isBreak[j] = false$. Если выполнено (2), то флаг поломки шарнира $isBreak[j] = true$ и в дальнейшем для данного шарнира больше не вычисляются импульсы реакций связей, что приводит к разрыву шарнира.

2. Моделирование отключения двигателей при разрыве шарнира

При разрыве шарнира некоторые двигатели робота теряют свою управляемость. Предполагается, что электропитание двигателей осуществляется по кабелю от источника питания, находящегося в корпусе робота. В качестве одного из вариантов поведения робота при поломке шарнира предлагается отключать только те двигатели, которые находятся в поддереве (из M шарниров) сломанного шарнира.

На рис. 1 изображено дерево, узлами которого являются шарниры с двигателями. При разрыве шарнира 5 будут отключены двигатели 6, 7 и 8. Отключение двигателя означает, что он перестает развивать момент, т.е. $\tau_m = 0$, но при этом ограничение для двигателя [5], связывающее скорость звена с угловой скоростью двигателя продолжает обрабатываться.

В некоторых роботах возможны механизмы, когда один двигатель управляет несколькими звеньями. Звено, которым управляет двигатель, называется основным, остальные управляемые звенья называются повторяемыми, а соответствующие шарниры повторителями. Примером механизма с повторителями является губки захвата манипулятора робота. В данном механизме двигатель управляет одновременно двумя губками, обеспечивая тем самым их плоскопараллельное движение.

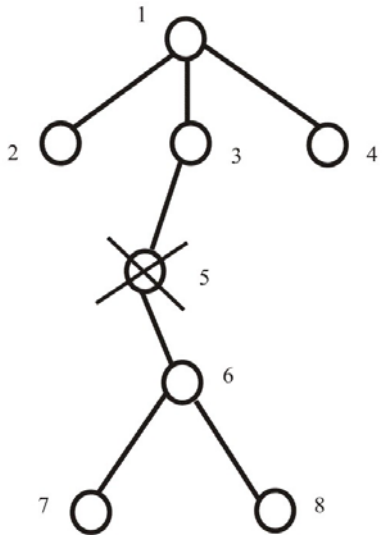


Рис. 1. Дерево шарниров робота

После разрыва шарнира возможны следующие ситуации взаимного расположения двигателя и повторителя:

1. Двигатель сломался или находится в оторванной группе, а повторитель находится в сохранившейся группе.
2. Двигатель находится в сохранившейся группе, а повторитель находится в оторванной группе.
3. Двигатель и повторитель находятся в оторванной группе.
4. Двигатель и повторитель находятся в сохранившейся группе.

В первых двух случаях соответствующее повторяемое звено перестает быть управляемым. В методе последовательных импульсов это означает, что ограничение для повторителя, связывающее скорость повторяемого звена с угловой скоростью двигателя, в дальнейшем перестает обрабатываться. Но при этом для обеспечения качественного моделирования требуется передать от двигателя к шарниру-повторителю максимальный момент сухого трения. В третьем случае двигатель и звено повторителя перестают быть управляемыми, но ограничение для повторителя сохраняется. В последнем случае никаких изменений для повторителя не потребуется.

Рассмотрим систему тел, в которой N шарниров, N_e двигателей и N_r повторителей. Пусть при поломке одного из шарниров в его поддереве содержится M шарниров. Алгоритм отключения двигателей после разрыва одного шарнира будет следующим:

1. Для каждого l -го повторителя ($l = 1,..., N_r$):
   $workRepeator[l] = true$
2. Для каждого j -го двигателя ($m = 1,..., N_e$):
   $workEngine[m] = true$
3. На i-м шаге моделирования:
   3.1. Если для s-го шарнира $isBreak[s] = true$, то
      3.1.1. Цикл по всем шарнирам в поддереве s-го шарнира ($k = 1,..., M$)
         Если k -ый шарнир является двигателем, то
            $workEngine[k] = false$
         Если k -ый шарнир является повторителем, то
            $workRepeator[k] = false$
      Конец цикла по k
      3.1.2. Цикл по всем шарнирам $j = 1,..., N$
         Если j -ый шарнир является повторителем k-го двигателя и $workEngine[k] = false$, то
            Если ($!workRepeator[j]$ && $!isBreak[k]$),
               то $workRepeator[l] = true$
            в противном случае
               $workRepeator[j] = false$
         Конец если
      Конец цикла по j
   Конец если

В данном алгоритме флаг $workRepeator[j]$ является индикатором того, что j-ый шарнир является повторителем двигателя. Если флаг $workRepeator[j]$ становится ложным, то для повторителя перестает обрабатываться ограничение, связывающее скорость звена с угловой скоростью двигателя. Флаг $workEngine[j]$ отвечает за работу j-го двигателя. Если флаг истинный, то двигатель создает момент и является управляемым, в противном случае двигатель не работает. На шагах 1 и 2 происходит инициализация данных флагов значением *true* перед началом моделирования системы тел.

Алгоритм состоит в том, чтобы на каждом шаге 3 моделирования динамики системы тел в случае поломки шарнира на шаге 3.1 необходимо отключить все двигатели, находящиеся в поддереве данного шарнира, и обработать все повторители в зависимости от их взаимного расположения с двигателями. В случае поломки шарнира на шаге 3.1.1 обрабатываются все шарниры, которые лежат ниже в иерархии. Если j-ый шарнир является повторителем, то сбрасывается флаг $workRepeator[j]$, а если двигателем, то сбрасывается флаг

$workEngine[j]$. Отметим, что на шаге 3.1.1 автоматически будет учтен 2-ой случай взаимного расположения двигателя и повторителя. На шаге 3.1.2 обрабатываются все шарниры. Если *j*-ый шарнир является повторителем *k*-го двигателя, который отключен, то возможны два варианта. Если повторитель отключен и двигатель не сломался, то это соответствует 3-му случаю взаимного расположения двигателя и повторителя, поэтому устанавливаем флаг $workRepeator[j] = true$. В противном случае это будет 1-ый случай взаимного расположения двигателя и повторителя, поэтому устанавливаем флаг $workRepeator[j] = false$.

3. Результаты моделирования

Предложенные алгоритмы и методы моделирования поломки шарниров были реализованы в подсистеме динамики имитационно-тренажерного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН.

Рис.2. Мобильный робот MF-4

.

Для апробации предложенных в статье алгоритмов был выбран мобильный гусеничный робот MF-4. На рис. 2 изображена его виртуальная модель, на которой обозначены два шарнира с номерами 1 и 2, для которых задаются параметры разрыва.

Для шарнира с номером 1 заданы следующие параметры: F_{max} = 30000 *H*, τ_{max} = 8000 $H \cdot м$, для шарнира с номером 2 заданы параметры: F_{max} = 30000 *H*, τ_{max} = 5000 $H \cdot м$.

Для моделирования было выбрано два сценария. В первом случае робот, разворачиваясь, ударяется об стенку (рис. 3), а во втором, находясь на подъемнике, падает с некоторой высоты (рис. 4). В первом случае разрывается только шарнир с номером 1, а во втором – оба шарнира.

После поломки шарниров теряется управление двигателями в сломавшейся части робота. В данных примерах была рассмотрена ситуация, когда двигатель, управляющий губками захвата манипулятора, оказался в оторванной части робота.

Результаты моделирования на примере виртуального робота MF-4 показывает применимость рассмотренных в статье алгоритмов и методов для обработки внештатных ситуаций.

Заключение

Предложенные в статье алгоритмы и методы позволяют моделировать возникновение внештатных ситуаций, связанных с поломкой мобильных роботов. Поведение виртуального робота, у которого ломаются шарниры и отключаются двигатели, позволяет наглядно продемонстрировать ошибки оператора тренажера в процессе управления роботом. Апробация рассмотренных в статье алгоритмов и методов показала их применимость для решения задач, связанных с управлением мобильными роботами в имитационно-тренажерных комплексах.

Исследование было выполнено при финансовой поддержке РФФИ в рамках научных проектов № 16-07-01104 и № 16-37-00107 мол_а.

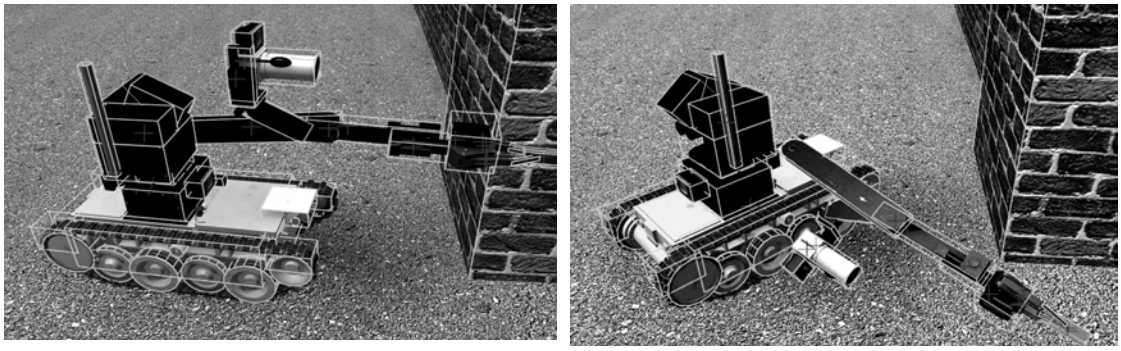


Рис. 3. Удар роботом MF-4 о стенку

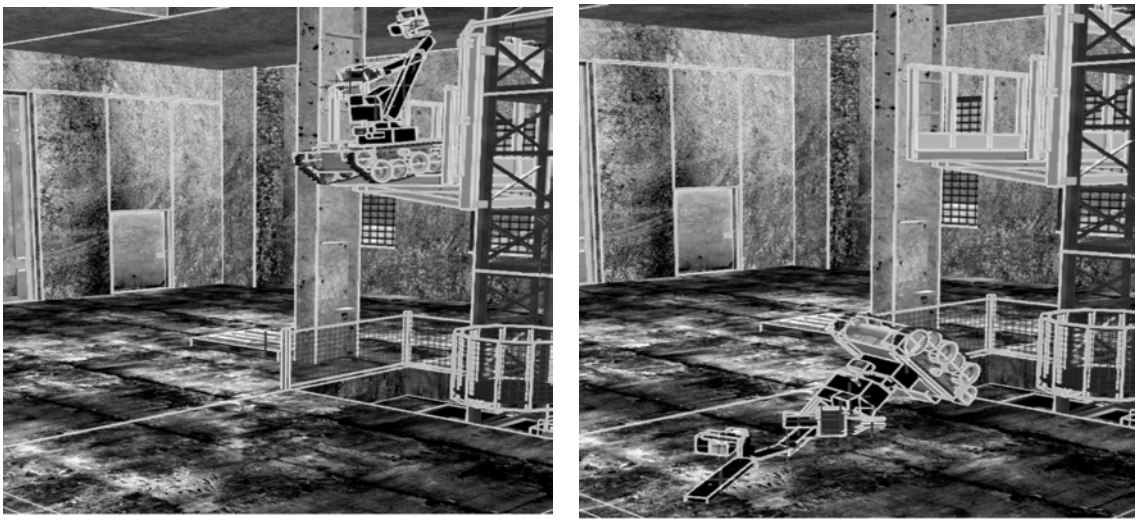


Рис. 4. Падение робота MF-4 с высоты

Simulation breaking joints of virtual robots

E.V. Strashnov, M.A. Torgashev

Abstract: The paper considers a problem of virtual robots behavior simulation in the extraordinary events with joints destruction under the action of external loads. With using sequential impulses an allowable load that the joint can undergo described with constraints imposed on accumulated impulses in the joints. In case of breaking joints of virtual robots is proposed to process an emergency situation with disabling some engines. Approbation of proposed algorithms and methods was carried out in the dynamic subsystem of training complex developed in SRISA RAS.

Keywords: joint destruction, extraordinary events, sequential impulses method, training complexes.

Литература

1. UNITY 5.3 Documentation. URL: https://docs.unity3d.com/ru/current/Manual/UnityManual.html (дата обращения: 27.01.17).
2. NVIDIA PhysX SDK 3.3.4 Documentation.
URL: http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Index.html
(дата обращения: 27.01.17).
3. https://ru.wikipedia.org/wiki/Ragdoll-физика (дата обращения: 27.01.17).
4. М.В. Михайлюк, Е.В. Страшнов. Моделирование системы связанных тел методом последовательных импульсов // Труды НИИСИ РАН, 2014, том 4, № 2, стр. 52-60.
5. Е.В.Страшнов, М.А.Торгашев. Моделирование динамики электроприводов виртуальных роботов в имитационно-тренажерных комплексах // Издательство “Новые технологии”, Мехатроника, автоматизация, управление, 2016, том 17, № 11, стр. 762-768.

Методы эргономичного управления объектами и параметрами виртуальной среды

А.В.Мальцев [1], М.В. Михайлюк [2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail's : [1] avmaltcev@mail.ru, [2] mix@niisi.ras.ru,

Аннотация: В работе рассматривается задача реализации эргономичного бесконтактного интерфейса управления объектами и параметрами трехмерных виртуальных сцен. Для ее решения предлагаются методы и алгоритмы, основанные на идентификации поз и жестов рук оператора с использованием устройства Microsoft Kinect. Подробно описаны методы кодирования и определения управляющих поз и жестов. Апробация предложенных в статье решений была проведена в составе имитационно-тренажерного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН.

Ключевые слова: трехмерная сцена, виртуальная среда, интерфейс управления, жест, поза, Kinect.

Введение

При создании систем виртуальной реальности и имитационно-тренажерных комплексов существенной составляющей является выбор интерфейсов управления объектами и параметрами трехмерного виртуального пространства, генерируемого с помощью современных компьютерных средств и технологий. Все интерфейсы подразделяются на две большие группы: контактные и бесконтактные. Первые подразумевают непосредственное прикосновение руками к элементам управления (например, клавиатура, "мышь", джойстик и т.п.). Вторые основаны на идентификации действий оператора, таких как произнесение речи, движения, жестикуляция и т.д.

Интерфейсы управления контактного типа, как правило, не являются интуитивно понятными и эргономичными при взаимодействии с виртуальной средой. Они требуют специального обучения оператора принципам работы с интерфейсом, но даже после обучения остаются неудобства в процессе его использования. На сегодняшний день преимущество всё больше переходит к бесконтактным технологиям управления. В качестве основы для многих бесконтактных подходов применяется устройство Microsoft Kinect, которое было разработано для игровой станции Xbox, но затем нашло применение и в области персональных компьютеров с операционной системой Windows. Популярность данного устройства в сфере создания бесконтактных интерфейсов взаимодействия человека с виртуальной средой отражается в большом числе российских и зарубежных публикаций. В статье [1], например, демонстрируется методы управления виртуальной моделью кисти руки с использованием Kinect. В работах [2, 3] рассматривается подход к управлению трехмерной виртуальной моделью антропоморфного робота в так называемом копирующем режиме, когда робот повторяет движения оператора. Публикации [4-6] описывают управление виртуальными объектами с помощью жестов, определяемых с использованием Kinect. В [7] устройство применяется для отслеживания положений кисти и пальцев руки человека. Полученные данные ассоциируются с различными командами работы с приложениями (открыть, закрыть и т.д.).

В данной работе предлагаются новые методы и алгоритмы реализации эргономичного бесконтактного интерфейса управления трехмерными объектами и параметрами виртуальной среды с помощью поз и жестов рук оператора. Разработанные решения основаны на использовании устройства Kinect для идентификации углов в сочленениях рук и формировании на их основе данных о текущей позе и жесте оператора. Также предлагаются эффективные методы кодирования жестов и поз в виде целочисленных идентификаторов. Далее рассмотрим эти подходы подробнее.

1. Определение и кодирование позы руки

Определим позу руки человека как совокупность значений углов в ее суставах и состояния S ладони. В данной работе используются углы φ и ψ в плечевом суставе, а также θ, γ – в локтевом (рис. 1). Будем считать все эти углы нулевыми в таком

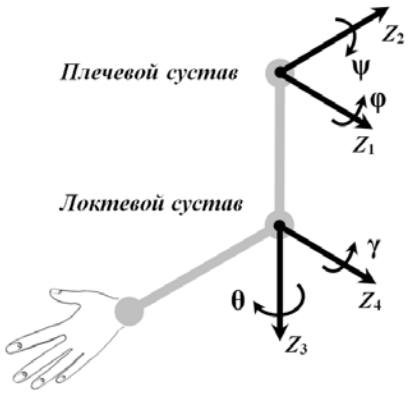


Рис. 1. Задание позы руки
по углам в ее суставах

положении руки, при котором плечо и предплечье перпендикулярны друг другу, плечо прижато к туловищу и параллельно ему, а предплечье направлено вперед. Угол поворота звена (плеча или предплечья) руки вокруг оси с направляющим вектором $\boldsymbol{Z}_i$ определим как положительный, когда вращение производится против часовой стрелки, если смотреть с конца вектора $\boldsymbol{Z}_i$ (рис. 1). Состояние S примем равным 1, если ладонь сжата в кулак, и 0 – в противном случае.

Значения параметров φ, ψ, θ, γ и S в текущий момент времени вычислим с помощью устройства Microsoft Kinect. Для этого представим плечевой и локтевой суставы в виде системы иерархически связанных осевых шарниров (рис. 2). Используя функции из Kinect API [8], получим координаты опорных точек скелета оператора, находящегося в рабочей области устройства. Для расчета искомых параметров на основе этих координат можно воспользоваться методами и алгоритмами подробно описанными в [2].

Поскольку хранение позы руки в виде набора углов и состояния ладони является затратным по памяти и неудобным при сравнении нескольких поз, мы предлагаем использовать цифровое кодирование. Подход состоит в представлении каждой позы в виде

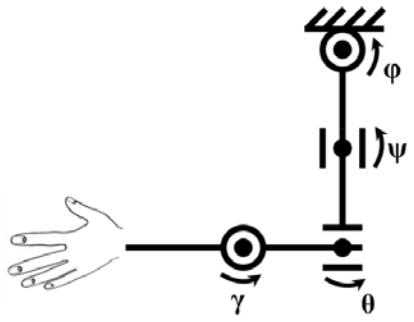


Рис. 2. Представление суставов
в виде иерархии шарниров

одного 32-битного беззнакового целого числа I_p – *идентификатора позы*, содержащего данные об углах φ, ψ, θ, γ и состоянии S. Непрерывный диапазон [-180°, 180°], охватывающий значения данных углов, разделим на открытые справа полуинтервалы одинаковой длины d_a, последовательно нумеруемые от 0 до $n = (360/d_a)-1$. Значение d_a выбирается, исходя из решаемой задачи, и составляет в среднем порядка 20°. Все углы, попадающие в один полуинтервал, будем считать эквивалентными значению его середины. Тогда любой угол $x \in$ [-180°, 180°) можно охарактеризовать идентификатором $I(x)$, равным номеру соответствующего этому углу полуинтервала:

$$I(x) = \left[\left(x + 180° \right) / d_a \right], \qquad (1)$$

где квадратные скобки обозначают целую часть числа. В связи с анатомией человеческой руки, значение 180° не достигается ни одним из углов φ, ψ, θ, γ, поэтому его можно не рассматривать. Фиксированная последовательность состояния S ладони (1 бит) и идентификаторов $I(\varphi)$, $I(\psi)$, $I(\theta)$, $I(\gamma)$ углов руки (7 бит на каждый) составляют идентификатор I_p позы руки. Вычисление значения такого идентификатора будем производить по формуле

$$\begin{aligned} I_p = &\left(S << 28 \right) + \left(I(\varphi) << 21 \right) + \\ &+ \left(I(\psi) << 14 \right) + \left(I(\theta) << 7 \right) + I(\gamma) \end{aligned}, \qquad (2)$$

где << обозначает операцию побитового сдвига влево.

При решении некоторых задач можно определить такие позы, для которых значения одного или нескольких углов будут неважны. Тогда во все разряды I_p, хранящие идентификаторы этих углов, будем записывать 1. Это можно сделать, выполнив операцию «побитовое ИЛИ» между I_p и целочисленной беззнаковой маской M_p, имеющей структуру эквивалентную I_p, но каждый значащий разряд которой будет содержать 0, а незначащий – 1. Для проведения корректного сравнения поз, маску M_p необходимо хранить вместе с соответствующим ей идентификатором позы.

Сравнение текущей позы $I_{p,cur}$ руки оператора, определяемой с помощью Kinect, с одной из заранее заданных поз I_p, имеющей маску M_p, будем производить путем проверки равенства

$$I_p = I_{p,cur} \mid M_p, \qquad (3)$$

где знак " | " обозначает операцию «побитовое ИЛИ». При соблюдении данного равенства позы $I_{p,cur}$ и I_p являются идентичными.

2. Кодирование и идентификация жестов

Любой жест руки оператора можно представить в виде некоторой конечной последовательности из N пройденных ею поз. Для простоты в данной работе ограничимся тремя позами. Идентификаторы I_p поз, входящих во все заранее выбранные управляющие жесты, вместе с соответствующими им масками M_p занесем в таблицу T поз, исключая повторение (рис. 3). Тогда любой из этих пар (I_p, M_p) будет соответствовать уникальный номер $T_p \in [0, m \cdot N\text{-}1]$ строки в таблице T, где m – общее количество заданных управляющих жестов.

Каждый из управляющих жестов будем кодировать в виде беззнакового целочисленного *идентификатора* I_g, содержащего в заданном порядке номера $T_{p,i}$ строк из таблицы

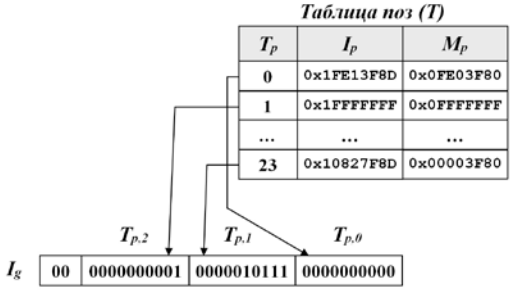


Рис. 3. Кодирование жеста с помощью целочисленного идентификатора

T, которые соответствуют входящим в этот жест позам ($i \in [0, 2]$ – очередность позы в жесте). В нашем случае I_g представляет собой 32-битное целое число, в котором под значение каждого $T_{p,i}$ отводится 10 бит (рис. 3). Вычисляется такой идентификатор по формуле

$$I_g = \sum_{i=0}^{2} \left(T_{p,i} << 10i \right). \qquad (4)$$

Для управления объектами и параметрами некоторой виртуальной среды необходимо заранее определить набор требуемых действий/операций и соответствующих им управляющих жестов, а также вычислить для каждого жеста его идентификатор I_g по формуле (4). Совокупность всех пар (I_g, операция) будет составлять *базу* B_g *управляющих жестов*.

Теперь рассмотрим определение жеста оператора, находящегося в рабочей области устройства Kinect. Пусть t_{max} – максимальное время, отведенное на один жест. Производя опрос устройства Kinect каждые Δt секунд (выполняется с использованием Kinect API), будем получать координаты опорных точек скелета оператора. На основе этих данных рассчитаем (см. [2]) значения углов φ, ψ, θ, γ в сочленениях его руки (или обеих рук). Используя формулу (2), вычислим идентификатор $I_{p,t}$ позы руки в текущий момент времени t. Если $I_{p,t}$ не совпадает с идентификатором $I_{p,t-\Delta t}$ позы той же руки в момент предыдущего опроса устройства, то рука движется, и необходимо начать отслеживание производимого ею жеста. При этом время начала жеста $t_0 = t$, первая поза жеста определяется как $I_{p,0} = I_{p,t-\Delta t}$, а вторая – $I_{p,1} = I_{p,t}$. Далее, продолжая опрашивать Kinect, каждый раз проверяем равенство текущей и предыдущей поз. Если в некоторый момент выполняется условие $t - t_0 \geq t_{max}$, то жест считается несостоявшимся и процесс начинается сначала.

Если $t - t_0 < t_{max}$ и $I_{p,t} \neq I_{p,t-\Delta t}$, то мы достигли финальной третьей позы жеста и $I_{p,2} = I_{p,t}$. Теперь, получив три идентификатора $I_{p,0}$, $I_{p,1}$, $I_{p,2}$, проведем поиск соответствующих им поз в таблице T. Для этого, пробегая таблицу сверху вниз, выполним сравнение идентификаторов по формуле (3). При нахождении в T всех поз выявленного жеста, сформируем на основе соответствующих им номеров $T_{p,i}$ 32-битный идентификатор I_g жеста, применяя формулу (4). При отсутствии какой-либо из поз в таблице T, установим все биты ее $T_{p,i}$ в 1. Если сформированный идентификатор I_g существует в заранее созданной базе B_g управляющих жестов, выполним приписанное к данному жесту действие.

3. Управление объектами и параметрами сцены

В качестве примера жесто-позового интерфейса в данной работе мы рассмотрим управление виртуальным наблюдателем (камеры), а также параметрами моделирования атмосферных осадков в сцене.

Тремя основными операциями виртуальной камеры являются движение, поворот и трансфокация. Движения осуществляются вдоль осей X, Y, Z локальной системы координат VCS самой камеры, а повороты вокруг этих осей. Трансфокация представляет собой изменение фокусного расстояния для масштабирования видимых посредством камеры объектов. Чтобы решить задачу эргономичного управления виртуальной камерой, предлагается использовать правую руку для выбора операции, производимой этой камерой, а также оси локальной системы VCS, которая участвует в данной операции. В таблице 1 представлены выполняемые камерой действия, используемые для них оси,

Таблица 1. Управляющие позы правой руки для движения и поворота камеры

Операция	Ось	Поза						I_p	M_p
		рука, ладонь	S	φ	ψ	θ	γ		
Движение	X	горизонтально вправо, разжата	0	-	-90°	-	90°	0x0FE13F8D	0x0FE03F80
	Y	вертикально вверх, разжата	0	-175°	0°	-	90°	0x00027F8D	0x00003F80
	Z	горизонтально вперед, разжата	0	-90°	0°	-	90°	0x00827F8D	0x00003F80
Поворот	X	горизонтально вправо, сжата	1	-	-90°	-	90°	0x1FE13F8D	0x0FE03F80
	Y	вертикально вверх, сжата	1	-175°	0°	-	90°	0x10027F8D	0x00003F80
	Z	горизонтально вперед, сжата	1	-90°	0°	-	90°	0x10827F8D	0x00003F80
Трансфокация	-	свободна, сжата	1	-	-	-	-	0x1FFFFFFF	0x0FFFFFFF

соответствующие им управляющие позы правой руки, а также идентификаторы I_p поз (вычисленные по формулам (1), (2) при d_a = 20°) и маски M_p этих поз.

Левая рука задает параметры выполняемой в текущий момент операции (скорость и направление движения, направление поворота, коэффициент трансфокации). Для применения этих параметров интерфейс хранит в оперативной памяти коэффициент k_t трансфокации камеры, а также матрицу M преобразования из мировой системы координат WCS в систему VCS, определяющую текущее положение и ориентацию камеры (в начальный момент матрица является идентичной, а k_t = 1.0). Увеличение k_t производится при сгибании ($\gamma < 0$), а уменьшение – при разгибании левой руки в локте ($\gamma > 0$) относительно исходного положения, представленного на рис. 1. При движении камеры, рассчитывается матрица M_T переноса относительно ее текущего положения на некоторую величину Δs вдоль оси системы VCS, определяемой позой правой руки. Значение Δs вычисляется, исходя из угла γ для движения вдоль осей Y, Z (если $\gamma < 0$, $\Delta s > 0$ для Y и $\Delta s < 0$ для Z, при $\gamma > 0$ – наоборот) или угла θ при движении вдоль X (если $\theta > 0$, $\Delta s > 0$ и наоборот). При $\gamma = 0$ или $\theta = 0$ движения виртуальной камеры вдоль соответствующих осей не осуществляются, т.е. $\Delta s = 0$.

Аналогично, в случае поворота формируется матрица M_R поворота на угол $\Delta\alpha$ вокруг нужной оси системы VCS. Значение $\Delta\alpha$ определяется, исходя из угла γ для поворота вокруг оси X (если $\gamma < 0$, $\Delta\alpha > 0$ и наоборот) или угла θ при повороте вокруг Y, Z (если $\theta < 0$, $\Delta\alpha > 0$ для Y и Z, при $\theta > 0$ – наоборот). При $\gamma = 0$ или $\theta = 0$ повороты виртуальной камеры вокруг соответствующих осей не осуществляются, т.е. $\Delta\alpha = 0$.

Применение движений и поворотов виртуальной камеры осуществляется путем пересчета матрицы M по формуле $M' = M_T^{-1} M_R^{-1} M$.

Для реализации жестового управления параметрами виртуальной среды, введем четыре универсальных жеста руки, начинающихся от позы I_p(0, 0°, 0°, 0°, 0°):

- $I_{g,up}$ – сгибание в локте до γ = -50°;
- $I_{g,down}$ – разгибание в локте до γ = 50°;
- $I_{g,left}$ – поворот в локте налево до θ = 50°;
- $I_{g,right}$ – поворот в локте направо до θ = -50°.

Все позы, входящие в эти жесты, отражены в таблице 2. Идентификаторы поз вычисляются по формулам (1), (2) при d_a = 20°.

Таблица 2. Позы управляющих жестов

T_p	Поза I_p	Маска M_p
0	I_p(0, 0°, 0°, 0°, 0°)	0x00000000
1	I_p(0, 0°, 0°, 0°, -25°)	0x00000000
2	I_p(0, 0°, 0°, 0°, -50°)	0x00000000
3	I_p(0, 0°, 0°, 0°, 25°)	0x00000000
4	I_p(0, 0°, 0°, 0°, 50°)	0x00000000
5	I_p(0, 0°, 0°, -25°, 0°)	0x00000000
6	I_p(0, 0°, 0°, -50°, 0°)	0x00000000
7	I_p(0, 0°, 0°, 25°, 0°)	0x00000000
8	I_p(0, 0°, 0°, 50°, 0°)	0x00000000

Значения идентификаторов $I_{g,up}$, $I_{g,down}$, $I_{g,left}$, $I_{g,right}$ получим, подставляя номера T_p строк таблицы, соответствующие одному жесту, в формулу (4).

Таблица 3. База управляющих поз и жестов

Операция	Поза левой руки	Жест правой руки
Начало дождя	I_p(0, 0°, 0°, 0°, 0°)	$I_{g,right}$

Усиление дождя	$I_p(0, 0°, 0°, 0°, 0°)$	$I_{g,up}$
Ослабление дождя	$I_p(0, 0°, 0°, 0°, 0°)$	$I_{g,down}$
Прекращение дождя	$I_p(0, 0°, 0°, 0°, 0°)$	$I_{g,left}$
Начало снега	$I_p(1, 0°, 0°, 0°, 0°)$	$I_{g,right}$
Усиление снега	$I_p(1, 0°, 0°, 0°, 0°)$	$I_{g,up}$
Ослабление снега	$I_p(1, 0°, 0°, 0°, 0°)$	$I_{g,down}$
Прекращение снега	$I_p(1, 0°, 0°, 0°, 0°)$	$I_{g,left}$

В таблице 3 представлены параметры моделируемых в виртуальной среде осадков и соответствующие им управляющие позы и жесты рук оператора.

Заключение

Апробация предложенных в статье решений была проведена в составе имитационно-тренажерного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН. Описанные методы реализации бесконтактного интерфейса позволяют расширить доступные средства эргономичного управления объектами как в системах виртуального окружения, так и в реальном пространстве.

Исследования были выполнены при финансовой поддержке РФФИ, грант № 15-07-04544.

Ergonomic control methods of objects and parameters in virtual environment

A.V. Maltsev, M.V. Mikhayluk

Abstract: At this paper a task of contactless interface realization to control objects and parameters of three-dimensional virtual scenes is considered. To solve it, methods and algorithms based on identification of an operator's hand postures and gestures by using the Microsoft Kinect device are presented. Methods for encoding and definition of control postures and gestures are described in detail. Approbation of proposed solutions was carried out as part of simulation training complex developed in SRISA RAS.

Keywords: three-dimensional scene, virtual environment, control interface, gesture, pose, Kinect.

Литература

1. I.Oikonomidis, N.Kyriazis, A.Argyros. Efficient model based 3D tracking of hand articulations using Kinect // In Proceedings of the 22-nd British Machine Vision Conference, 2011, p. 101.1-101.11.
2. А.В.Мальцев, М.В.Михайлюк. Реализация эргономичного интерфейса управления виртуальной моделью антропоморфного робота с использованием Kinect // Программная инженерия, 2015, № 10, с. 12-18.
3. А.И.Лапта, А.В.Мальцев, М.В.Михайлюк. Эргономичный интерфейс управления антропоморфным роботом // Труды 11-й Международной научно-практической конференции «Пилотируемые полеты в космос». – 2015. – С. 245-247.
4. J.O.Kim, M.Kim, K.H.Yoo. Real-Time Hand Gesture-Based Interaction with Objects in 3D Virtual Environments // International Journal of Multimedia and Ubiquitous Engineering, 2013, vol. 8, № 6, p. 339-348.
5. J.Soh, Y.J.Choi, Y.Park, H.S.Yang. User-friendly 3D object manipulation gesture using kinect // Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry, 2013, p. 231-234.
6. K.Qian, J.Niu, H.Yang. Developing a Gesture Based Remote Human-Robot Interaction System Using Kinect. International Journal of Smart Home, 2013, vol. 7, no. 4, p. 203-208.
7. Л.А.Котюжанский. Интерфейс бесконтактного управления // Фундаментальные исследования, 2013, №4, с. 44-48.
8. Kinect API Overview. URL: https://msdn.microsoft.com/ru-ru/library/dn782033.aspx (дата обращения: 13.02.2017).

Нечёткие меры и их использование в оценке алгоритмов компьютерного зрения

А.С. Осипов

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,
E-mail: osipa@niisi.ras.ru

Аннотация: В статье рассматриваются вопросы, связанные с исследованием производительности алгоритмов компьютерного зрения, в рамках разработанного в НИИСИ РАН эмпирического подхода к их оценке. Данный подход использует элементы нечёткой логики, в частности, нечёткие меры оценки качества алгоритмов. Рассматривается ряд известных мер оценки качества сегментации изображений и предлагается их нечёткий аналог. Применительно к задаче оценки алгоритмов распознавания лиц, вводятся новые нечёткие меры и рассматриваются вопросы их практического использования.

Ключевые слова: оценка производительности, сегментация изображений, распознавание лиц, ground truth образы, теория нечётких множеств, нечёткие меры сходства.

Введение

В последние десятилетия в области компьютерного зрения разработано и используется большое число алгоритмов, в той или иной степени использующих элементы теории нечётких множеств [1-2]. При решении ряда задач обработки и анализа изображений использование таких алгоритмов представляется вполне оправданным, как на стадии обработки (выделение границ, сегментация и т. д.), так и на последующих стадиях (классификация и распознавание). Например, как справедливо отмечалось в [3], «полутоновые изображения являются нечёткими по своей природе из-за неопределённости, существующей в локализации положения границы, отделяющей объект от фона». Особенно это характерно для размытых изображений. Классификация объектов, полученных после стадии обработки изображений (например, в задаче распознавания букв в тексте) также не всегда является однозначной. Возникает естественная задача сравнительной оценки упомянутых выше алгоритмов, предназначенных для решения одной и той же задачи [4]. Требуется уметь сравнивать эти алгоритмы, как между собой, так и с алгоритмами, не использующими нечёткой логики.

Следует отметить, что оценка качества работы различных компьютерных программ, решающих некоторую практическую задачу, представляет собой процесс, не имеющий единой методики. Основные отличия методик, применяемых в сравнительных исследованиях различных программ, заключаются в следующем:

- тип критерия оценки качества (количественный или качественный, использующий эталоны или нет);
- тип эталонов и тестовых материалов (реальные или синтезированные), их параметры, количество, источники (оригинальные или общедоступные) и т.п.

Разрабатываемая в ФГУ ФНЦ НИИСИ РАН методика строит оценку качества на основе вычисления некоторой количественной меры сходства, оценивающей результат работы программы на некотором наборе тестовых изображений, для которых эталонное решение, так называемое ground truth, известно априори. Такой подход к оценке качества программных продуктов в англоязычной литературе называется discrepancy method, что определило выбор названия нашей методики: EDEM (Empirical Discrepancy Evaluation Method) [4-6]. Таким образом, можно отнести данную методику к классу контролируемых эмпирических методик оценки тестируемых алгоритмов компьютерного зрения (подробнее см. [5]). Характерной чертой нашей методики является внесение в тестовые изображения контролируемых искажений: (с известными параметрами) и различных преобразований объектов на изображении. Данная методика находится в процессе совершенствования, как в плане выбора и построения тестовых материалов и соответствующих эталонов, так и в плане анализа и сочетания разных количественных мер сходства. Инструментальной средой поддержки методики EDEM является программная система PICASSO [4],[6]. Предназначение данной системы - предоставить возможность пользователю создавать, хранить и редактировать тестовые и эталонные изображения, вносить контролируемые возмущения в тестовый материал, использовать различные меры сходства, и проводить сравнительное тестирование различных

программных средств в интерактивном режиме и отображать результаты тестирования.

В рамках разрабатываемой нами методики, элементы нечёткой логики были первоначально применены к оценке алгоритмов обработки. Так, в [7] было предложено применить нечёткие ground truth эталоны и нечёткие меры сходства из работы [8] к оценке производительности детекторов границ. Введение нечётких ground truth эталонов преследовало цель не только использовать их для тестирования нечётких детекторов границ (которые могут присваивать каждому обработанному пикселю значения степени принадлежности нескольким классам: классам граничных пикселей, фону и шуму). Преследовалась также цель сделать процесс тестирования обычных «чётких» детекторов границ более глубоким. Именно, характерная особенность предложенного подхода заключалось в том, что одному тестовому изображению могло соответствовать несколько нечётких ground truth эталонов, при этом каждый из них предназначен для выявления некоторой особенности тестируемого алгоритма. Например, один нечёткий ground truth образ лучше определяет способность детектора границ выявлять слабоконтрастные края, в то время как другой образ более приспособлен для проверки способности детектора к выделению непрерывных границ. Таким образом, данный подход позволял сравнивать между собой качество работы как «чётких» так и «нечётких» детекторов границ, при этом сам метод оценки включал в себя элементы нечёткой логики. Эти две черты стали универсальными в нашей методике сравнительного исследования алгоритмов компьютерного зрения.

В работах [4],[5] рассматривалась возможность использования данного подхода при сравнительном исследовании алгоритмов сегментации изображений.

Что касается задач классификации и распознавания образов, в [6] исследовалась возможность применения данной методики к оценке алгоритмов распознавания человеческих лиц. Здесь следует отметить, что, например, большая вариабельность распознаваемых объектов свидетельствует о перспективности использования «нечётких» алгоритмов для решения данной задачи [9]. Следовательно, предлагаемая методика имеет здесь свою актуальность. В работе [6] был предложен новый метод построения нечётких ground truth эталонов для тестирования ряда методов распознавания лиц, использующих алгоритмы машинного обучения (подробнее см. ниже в разделе 4). Для оценки производительности данных методов использовались нечёткие меры сходства из работы [8]. При этом был сделан вывод, что для повышения качества тестирования необходима разработка новых нечётких мер сходства, чувствительных к той или иной особенности тестируемых алгоритмов. Рассмотрению данного вопроса в основном посвящена настоящая работа.

Статья организована следующим образом: в разделе 1 содержится краткий обзор количественных критериев, используемых в оценке алгоритмов сегментации изображений. В следующем разделе приводятся необходимые в дальнейшем понятия теории нечётких множеств и рассматриваются нечёткие аналоги некоторых мер близости, упомянутых в предыдущем разделе. Далее, в разделе 4 исследуется задача тестирования методов распознавания лиц с использованием разработанной нами методики. При этом, наряду с мерами сходства, использованными ранее в [6], [7] вводятся в рассмотрение новые нечёткие меры.

1. Меры оценки качества сегментации

Сегментация изображений – фундаментальная процедура в компьютерном зрении, являющаяся одним из этапов анализа изображения. Её цель – разбиение изображения на ряд непересекающихся областей-сегментов, однородных по некоторому признаку (например, уровень яркости, текстура, цвет) и существенно отличающихся по этому признаку от соседних (смежных) областей (подробнее см. [5]). Повышенный интерес научного сообщества к этой тематике выражается в быстро растущем числе публикаций, посвященных разработке, модернизации и применению различных алгоритмов сегментации изображений. В результате перед разработчиками систем компьютерного зрения возникает сложная задача выбора наиболее адекватных их задачам алгоритмов из множества имеющихся в литературе.

Эмпирические методики оценки алгоритмов компьютерного зрения оценивают не сам алгоритм, а результаты его работы на некотором наборе тестовых изображений. Они подразделяются на контролируемые и неконтролируемые (автоматические) методики. Разрабатываемую нами методику можно отнести к первому из данных классов.

В контролируемых эмпирических методиках оценки алгоритмов сегментации используются количественные меры близости между результатом работы алгоритма и эталонной сегментацией исходного изображения. Применительно к задаче оценки качества сегментации, данная эталонная сегментация представляет собой ground truth образ. Эта эталонная сегментация создаётся экспертом вручную или получается автоматически при генерации синтетического изображения [5].

В рамках данных методик, самые простые меры качества сегментации, которые сразу же начали использовать исследователи – это количество (или процент) пикселей, отнесенных при сегментации не к своему сегменту (неправильно классифицированных пикселей), или, напротив, процент правильно классифицированных пикселей [9],[10].

Так, например, в работе [10] было предложено два критерия, являющиеся обобщением для случая нескольких классов ошибок 1-го и 2-го рода, использующихся в задачах бинарной классификации.

Оба этих критерия основаны на построении матрицы неточностей (confusion matrix). Столбец этой матрицы соответствует классу, к которому пиксели принадлежат на самом деле, а строка – классу, к которому пиксели отнесены при сегментации. Правильно классифицированные пиксели относятся к элементам матрицы, находящимся на главной диагонали, неправильно классифицированные – ко всем остальным.

Первый из предложенных критериев – это процентное отношение неправильно классифицированных пикселей данного *k*-го класса к общему количеству пикселей этого класса на эталонном изображении:

$$M_1^k = \frac{\left(\sum_{i=1}^{n} C_{ik}\right) - C_{kk}}{\sum_{i=1}^{n} C_{ik}}, \qquad (1)$$

где n – количество классов,

C_{kk} – количество правильно классифицированных пикселей *k*-го класса,

$\sum_{i=1}^{n} C_{ik}$ – количество пикселей, в действительности принадлежащих к *k*-му классу.

Второй критерий – процентное отношение пикселей, ошибочно причисленных к данному *k*-му классу, к общему количеству пикселей других классов на эталонном изображении:

$$M_2^k = \frac{\left(\sum_{i=1}^{n} C_{ki}\right) - C_{kk}}{\left(\sum_{i=1}^{n}\sum_{k=1}^{n} C_{ik}\right) - \sum_{i=1}^{n} C_{ik}}, \qquad (2)$$

где $\sum_{i=1}^{n} C_{ki}$ – количество пикселей, отнесенных к *k*-му классу при сегментации,

$\sum_{i=1}^{n}\sum_{k=1}^{n} C_{ik}$ – общее количество пикселей на изображении,

n, C_{kk}, $\sum_{i=1}^{n} C_{ik}$ имеют тот же смысл, что и в (1).

Таким образом, при наличии n сегментов изображения получаем $2n$ критериев M_1^k, M_2^k, $k=1,2\ldots n$, позволяющих проанализировать вклад каждого сегмента в общую ошибку. Кроме того, в [10] было отмечено, что элементы матрицы неточностей могут быть взвешены с целью учесть разную значимость ошибок для разных сегментов изображения. Следует отметить, что и в ряде других подобных работ упоминалось о такой возможности, при этом конкретная процедура взвешивания нигде не приводилась. Разрабатываемый нами «нечёткий» метод оценки позволяет естественным образом реализовать данную процедуру.

В работе [9] упоминаются следующие четыре меры оценки качества сегментации: меры изготовителя (англ. producer's accuracy measure), пользователя (user's accuracy measure); меры Хеллдена и Шорта (Hellden's and Short's accuracy measures). Эти меры введены в рассмотрение в работах, посвящённых обработке изображений, полученных при дистанционном зондировании Земли [9]. Они сходны с мерами, определёнными в (1)-(2), так, для первой из них, меры изготовителя *PA*, справедливы соотношения:

$$PA^k = 1 - M_1^k, \qquad k = 1,\ldots,n.$$

Значения мер Хеллдена *HA* и Шорта *SA* определяются по формулам:

$$HA^k = \frac{2C_{kk}}{\sum_{i=1}^{n} C_{ik} + \sum_{i=1}^{n} C_{ki}}, \qquad (3)$$

$$SA^k = \frac{C_{kk}}{\sum_{i=1}^{n} C_{ik} + \sum_{i=1}^{n} C_{ki} - C_{kk}}, k = 1,.,n \qquad (4)$$

(нечёткие версии этих мер будут рассмотрены в следующих разделах).

Все вышеприведённые критерии оценки качества сегментации относятся к разряду статистических мер, т. е. они основаны исключительно на подсчёте неправильно (или правильно) классифицированных пикселей и имеют присущие таким мерам системные недостатки:

- возможность расхождения количественных результатов оценки со здравым смыслом, т. е. иногда результаты сегментации, явно лучшие с точки зрения экспертов, имеют более высокий процент ошибочно классифицированных пикселей;
- статистические меры не учитывают расположение ошибочных пикселей относительно соответствующего сегмента – с очевидно, что ошибка на границе и ошибка в центре сегмента должны штрафоваться по разному.

Поэтому, наряду со статистическими характеристиками, оценка месторасположения ошибочно классифицированных пикселей на эталонном отсегментированном изображении (ground truth образе) относительно сегмента, к которому они были ошибочно отнесены при сегментации, является важной характеристикой тестируемого алгоритма. Соответствующие меры получили название мер оценки локализации (англ. localization performance, or “distance” measures) [4].

Одна из первых таких мер, возникающая из общих соображений, была предложена в упоминавшейся выше работе [10]:

$$\varepsilon = \frac{\sqrt{\sum_{i=1}^{N} d_i^2}}{A},$$

где N – количество ошибочно классифицированных пикселей,

A – общее количество пикселей в изображении,

d_i – евклидово расстояние между i-ым ошибочно классифицированным пикселем x и ближайшим пикселем y, действительно относящимся к данному классу.

Мера ε может быть вычислена для всех вариантов ошибок (пиксель i-го класса ошибочно отнесен к j-му) и представлен в виде матрицы $K \times K$, где K – число классов. Элементы матрицы могут быть взвешены с целью учесть разную значимость разных вариантов ошибок (как и в случае статистических мер, соответствующая процедура в [10] предложена не была). При этом, в [10] подчёркивалась важность такой процедуры для повышения адекватности итоговой количественной оценки. Дело в том, что меры, введённые в [10] использовались в оценке качества сегментации изображений клеток крови. Естественно, что ошибки в определении границы клетки или границы ее ядра должны иметь больший вес в сравнении с ошибками классификации их внутренних частей.

В работе [11] широко известный критерий Прэтта (Pratt's figure of merit), предназначенный для оценки производительности детекторов границ (см., например, [4]) был адаптирован для целей оценки качества сегментации. Введенные в указанной статье меры качества сегментации FOM и FOM_e, так же, как и рассмотренный выше критерий ε, основаны на определении того, насколько далеко данный пиксель отстоит от «правильного» местоположения пикселя его класса:

$$FOM = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + \gamma d_i^2}, \qquad (5)$$

где N – число пикселей в изображении,

d_i – расстояние от i-го пикселя изображения до ближайшего пикселя, отнесенного к тому же классу на эталонном изображении,

γ – масштабный множитель.

При полностью корректной сегментации FOM=1. На практике большинство элементов изображения относятся при сегментации к правильной области, и значения FOM попадают в довольно узкий интервал, близкий к 1 (подробнее см. [5]). Чтобы увеличить этот интервал и, тем самым, повысить степень наглядности сравнительной оценки разных алгоритмов, дающих хорошую сегментацию, при вычислении меры качества можно принимать во внимание только ошибочно классифицированные пиксели N_e:

$$FOM_e = \begin{cases} \frac{1}{N_e} \sum_{i=1}^{N_e} \frac{1}{1 + \gamma d_i^2}, & N_e > 0 \\ 1, & N_e = 0 \end{cases} \qquad (6)$$

При хорошей сегментации эталонное и результирующее изображения должны иметь одинаковую степень фрагментации, то есть количество сегментов на них должно совпадать (или почти совпадать).Поэтому, в той же работе [11], для оценки степени фрагментации изображения было предложено использовать следующую меру:

$$FRAG = \frac{1}{1 + \left|\alpha (n_R - n_I)\right|^{\beta}}, \qquad (7)$$

где n_R – количество сегментов на результирующем изображении,

n_I – количество сегментов на эталонном изображении,

α, β – масштабные параметры.

Чем ближе значение меры фрагментации к 1, тем выше оценка качества сегментации. Параметр α определяет вклад величины (n_R–n_I) в значение *FRAG*, а параметр β определяет, насколько сильно штрафуются большие отклонения n_R от n_I, по сравнению с малыми. Сами авторы использовали в своих экспериментах следующие значения масштабных параметров: α=0.16, β=2.

Как известно, сегментация является этапом, предшествующим анализу изображения. Формально её цель – представить изображение в виде, удобном для определения его количественных характеристик, используемых в дальнейшем для анализа этого изображения. С этой точки зрения, логично оценивать качество сегментации по тому, насколько точно (по сравнению с эталонным изображением) можно на сегментированном изображении определить такие характеристики. Поэтому, наряду со статистическими мерами и мерами оценки локализации, при исследовании методов сегментации используются и меры оценки этих характеристик.

Так, в работе [11], наряду с (5)-(7), была предложена мера *FOC* (figure of certainty), при вычислении которой оценивается такая характеристика как интенсивность:

$$FOC = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{1+\left|\psi(f_i-\mu_j)\right|^{\delta}}, \qquad (8)$$

где N – количество пикселей в изображении,

f_i – значение интенсивности пикселя i исходного изображения,

μ_j – репрезентативное значение интенсивности j-го сегмента, к которому i-й пиксель был отнесен при сегментации,

ψ, δ – масштабные параметры.

В простейшем случае, когда исходное изображение состоит из нескольких областей постоянной интенсивности, выражение $(f_i - \mu_j)$ представляет собой разность интенсивностей того сегмента, к которому i-й пиксель принадлежит на самом деле, и того, к которому он был отнесен в процессе сегментации. Таким образом, ошибка при сегментации штрафуется тем больше, чем значительнее такая разность. При идеальной сегментации *FOC*=1.

В более общем случае, когда изображение содержит неоднородные по интенсивности области и сегментация осуществляется на основе некоторого признака F, f_i будет значением этого признака в пикселе i, а в качестве μ_j можно использовать среднее значение признака F для j-го сегмента.

Масштабные параметры ψ и δ имеют тот же смысл, что и, соответственно, параметры α и β в (7), и их значения в [11] были одни и те же.

В статье [12] было предложено два критерия, основанных на оценке характеристик сегментов – *AUMA* (absolute ultimate measurement accuracy) и *RUMA* (relative ultimate measurement accuracy):

$$AUMA_f = \left|R_f - S_f\right|,$$

$$RUMA_f = \frac{\left|R_f - S_f\right|}{R_f}\times 100,$$

где R_f – значение признака f, полученное на эталонном изображении,

S_f – значение признака f, полученное на сегментированном изображении.

По существу, мы имеем не 2, а $2N$ критериев оценки, где N – количество выделяемых признаков. Очевидно, что чем ближе значения *AUMA* и *RUMA* к нулю, тем выше оценка качества сегментации. Обычно оценивают геометрические характеристики сегментов изображения (площадь, периметр, эксцентриситет и др.).

Следует отметить, что данные критерии оценивают общие характеристики сегментов, в то время как меры, подобные *FOC*, оценивают отличия значений характеристики исходного изображения в каждом пикселе (в случае *FOC* это яркость) от её репрезентативных значений для сегмента, к которому этот пиксель отнесен. В этом – главное отличие этих групп критериев.

Существуют также комплексные меры оценки качества сегментации, например, оценивающие как степень фрагментации изображения, так и геометрические характеристики сегментов (подробнее см. [5]).

2. О нечётких мерах оценки

Прежде всего, напомним несколько основных понятий из теории нечётких множеств.

Именно, пусть X есть непустое множество (например, множество пикселей изображения). Нечёткое множество C на X есть пара $< X, f_C >$, где f_C есть отображение X на [0, 1]. Значение $f_C(x)$ для элемента $x \in X$ называется степенью принадлежности x множеству C, (например, степень принадлежности данного пикселя тестового образа некоторому сегменту

эталонного ground truth образа) а функция f_C называется функцией принадлежности нечёткого множества. Нечёткое множество называется непустым, если хотя бы для одного элемента $x \in X$, $f_C(x)>0$.

Заметим, что обычные (чёткие) подмножества M из X включаются в данный подход, если мы будем рассматривать их как стандартные характеристические функции $1_M : X \to [0,1]$. То есть, например, если у нас имеется пиксель x, относящийся к некоторому классу (например, сегменту) C, то в этом случае $f_C(x) = 1$ и $f_{C1}(x) = 0$ для всех классов $C1$ отличных от C.

Будем предполагать, что множество X конечно.

Определение. *Нечёткой классификацией F множества X*

$$F := < X, f_{C_1},..., f_{C_N} >, \quad f_{C_m} : X \to [0,1], \quad m = 1,..., N.$$

называется совокупность N нечётких классов, удовлетворяющих условию:

$$\sum_{m=1}^{N} f_{C_m}(x) = 1 \qquad \forall x \in X.$$

Функция f_{C_m} - степень принадлежности соответствующему классу. Легко видеть, что обычная классификация X (т. е. разбиение X на N непересекающихся подмножеств-классов) является нечёткой классификацией (каждый элемент X принадлежит ровно одному классу), и в роли функций данной классификации выступают характеристические функции классов.

Обозначим множество всех нечётких множеств на X за $[0,1]^X$. Для A и B из $[0,1]^X$ нечёткое отношение включения

$A \in B$ означает что $f_A(x) \le f_B(x)$ для всех $x \in X$.

Нечёткая мера сходства есть отображение $s : [0,1]^X \times [0,1]^X \to [0,1]$, сопоставляющее множествам $A, B \in [0,1]^X$ степень сходства $s(A,B) \in [0,1]$, удовлетворяющее условиям:

- $s(A,A) = 1$ для любого нечёткого A
- $s(A,B) = s(B,A)$ для всех нечётких A и B (коммутативность)
- $s(A,C) \le s(A,B) \wedge s(B,C)$

при $A \subset B \subset C$, где $p \wedge q$ обозначает минимум из p и q; максимум из p и q обозначается как $p \vee q$. Важными для дальнейших рассмотрений примерами данных мер являются:

$$S_1(A,B) = \frac{\sum_{x \in X} f_A(x) \wedge f_B(x)}{\sum_{x \in X} f_A(x) \vee f_B(x)}, \qquad (9)$$

$$S_2(A,B) = \frac{\sum_{x \in X} 2(f_A(x) \wedge f_B(x))}{\sum_{x \in X} f_A(x) + f_B(x)}. \qquad (10)$$

Пусть имеется две обычных классификации $X: [X_1^1,..., X_N^1]$, $[X_1^2,..., X_N^2]$. Им соответствуют нечёткие классификации

$$F^1 = < X, f_1^1,..., f_N^1 >, \; F^2 = < X, f_1^2,..., f_N^2 >;$$
$$f_m^i = 1_{X_m^i}, \quad i = 1,2; \quad m = 1,...N.$$

Для числа общих элементов, принадлежащих классам из разных классификаций, имеем соотношения:

$$C_{ik} = | X_k^1 \cap X_i^2 | = \sum_{x \in X} f_k^1(x) \wedge f_i^2(x), \; i,k = 1,..., N.$$

Далее, вычислим значение $S_1(X_k^1, X_k^2)$:

$$S_1(X_k^1, X_k^2) = \frac{\sum_{x \in X} f_k^1(x) \wedge f_k^2(x)}{\sum_{x \in X} f_k^1(x) \vee f_k^2(x)} = \frac{C_{kk}}{\sum_{x \in X} f_k^1(x) \vee f_k^2(x)}.$$

Пользуясь очевидным числовым соотношением $\alpha \vee \beta = \alpha + \beta - \alpha \wedge \beta$ а также легко проверяемыми представлениями: $X_k^1 = \bigcup_{i=1}^{N} (X_k^1 \bigcap X_i^2)$, $X_k^2 = \bigcup_{i=1}^{N} (X_i^1 \bigcap X_k^2)$, преобразуем последнюю формулу к виду

$$S_1\left(X_k^1, X_k^2\right) = \frac{C_{kk}}{\sum_{x\in X} f_k^1(x) + \sum_{x\in X} f_k^2(x) - C_{kk}} = \frac{C_{kk}}{\sum_{i=1}^{N} C_{ik} + \sum_{i=1}^{N} C_{ki} - C_{kk}}.$$

Таким образом, мы получили значение, совпадающее со значением формулы (4). Иными словами, для обычных («чётких») множеств значения меры близости S_1 совпадают с со значениями меры оценки качества классификации Шорта. Аналогичное совпадение имеет место для S_2 и меры Хеллдена. Следовательно, можно считать, что S_1 и S_2 являются обобщением мер Шорта и Хеллдена на случай нечётких множеств.

На базе мер (9)-(10), в работе [8] также введены итоговые (по всем классам) меры сходства (overall accuracy measures):

$$OA_1\left(F^1, F^2\right) = \frac{\sum_i \sum_{x\in X} f_{C_i}^1(x) \wedge f_{C_i}^2(x)}{\sum_i \sum_{x\in X} f_{C_i}^1(x) \vee f_{C_i}^2(x)}, \quad (11)$$

$$OA_2\left(F^1, F^2\right) = = \frac{\sum_i \sum_{x\in X} 2(f_{C_i}^1(x) \wedge f_{C_i}^2(x))}{\sum_i \sum_{x\in X} f_{C_i}^1(x) + f_{C_i}^2(x)}. \quad (12)$$

Что касается нечётких аналогов мер Прэтта *FOM* и FOM_e, то здесь представляется естественным следующий вариант. Пусть есть нечёткая классификация X: $F = < X, f_1, \ldots, f_N >$. Будем считать, что элемент $x \in X$ относится к классу k в данной классификации, если степень принадлежности элемента x данному классу максимальна: $f_k(x) = \max_{1\le m\le N} f_m(x)$. Далее, пусть есть две нечётких классификации X:

$$F = < X, f_1, \ldots, f_N >, \; G = < X, g_1, \ldots, g_N >;$$

при этом G - эталонная (ground truth) классификация, а F – тестовая классификация (например, результат работы нечёткого метода сегментации). Назовём нечёткой мерой Прэтта *FFOM* величину

$$FFOM = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + \gamma d_i^2},$$

где d_i – расстояние от i-го элемента множества X, отнесённого к некоторому классу k в классификации G до ближайшего элемента X, отнесенного к тому же классу k в классификации F. Соответственно, N – число элементов X, γ – масштабный множитель.

Аналогичным способом определяется $FFOM_e$ – нечёткий аналог FOM_e (здесь ошибочными являются элементы множества X, отнесённые к разным классам в классификациях G и F). Легко видеть, что при оценке обычных («чётких») методов сегментации изображений, значения *FFOM* и $FFOM_e$ совпадают со значениями *FOM* и FOM_e (5)-(6), а значит, меры *FFOM* и $FFOM_e$ представляют собой обобщения мер Прэтта.

Мера *FRAG* (7) может применяться и при оценке нечётких алгоритмов, в особенности, когда число классов (например, сегментов изображения) в сравниваемых нечётких классификациях F и G множества X различно (в противном случае значение *FRAG* равно 1).

Использование меры *FOC* (8) также допустимо в «нечётком» случае. Например, при оценке нечётких алгоритмов сегментации, где параметр μ_j – репрезентативное значение интенсивности j-го сегмента, к которому i-й пиксель был отнесен при сегментации. При этом j-й сегмент состоит из пикселей, степень принадлежности которых j-му классу в классификации F (результату работы тестируемого алгоритма) максимальна.

Упомянутые выше критерии *AUMA* и *RUMA* можно применять к оценке качества нечётких алгоритмов сегментации. При этом принципы формирования сегментов определяют тестируемый алгоритм и эталонная сегментация.

3. Нечёткие меры в оценке алгоритмов распознавания лиц

Распознавание лиц – одна из типичных задач классификации и распознавания образов. Под задачей распознавания лиц (face recognition) обычно понимается следующее: дана база изображений (фотографий или видеокадров) конкретных людей, состоящая из конечного набора классов. Каждый класс изображений представляет собой изображения одного человека. По предъявлении входного изображения человеческого лица требуется определить его

принадлежность одному из данных классов (или установить отсутствие такой принадлежности).

Актуальность задачи распознавания лиц привела к появлению разных алгоритмов ее решения, следовательно, актуальным является и умение сравнивать эти алгоритмы между собой.

Как правило, образы базы изображений представляют собой обучающую выборку, при помощи которой определяются исходные параметры алгоритма. Затем, в качестве первичной верификации, к распознаванию предъявляется набор изображений тех же людей, чьи изображения содержатся в базе. Этот набор изображений составляет тестовую выборку.

Несмотря на обилие алгоритмов распознавания лиц, обычно эти алгоритмы реализованы по следующей схеме. На первом этапе решения данной задачи, для каждого изображения обучающей выборки составляется набор (вектор) его характерных признаков. На следующем этапе, используя векторы характерных признаков, с помощью алгоритмов машинного обучения строится модель - классификатор, эффективно разделяющая между собой наборы признаков, соответствующие изображениям разных лиц. Этот этап завершает первичную настройку алгоритма. Далее, для тестового изображения составляется вектор характерных признаков, который подаётся на вход классификатора, сравнивается с его содержимым, после чего делается заключение о принадлежности входного вектора одному из классов обучающей выборки.

В рамках данного подхода, наиболее естественным методом классификации изображений является метод поиска ближайшего соседа в пространстве образов. Каждый образ тестовой выборки представляет собой элемент в пространстве образов и ищется ближайшее расстояние между ним и элементами обучающей выборки. Элемент тестовой выборки относят к тому же классу, что и элемент обучающей выборки, на котором достигается это расстояние. Для эффективного распознавания весьма существенно, чтобы образы, соответствующие одному классу, располагались плотно, в кластерах, и при этом кластеры, соответствующие разным классам, были разделены между собой.

Весьма распространённым методом классификации является также метод k ближайших соседей (k -nearest neighbors algorithm) в соответствии с которым объект относят тому классу, который является наиболее распространённым среди соседей данного элемента. В качестве определения расстояния между объектами может использоваться любая общеупотребительная метрика (например, Евклидова). В задаче распознавания лиц используется и нечёткая версия данного алгоритма. В данной версии вначале для всех элементов обучающей выборки определяется степень их принадлежности классам базы изображений (чем больше из k ближайших соседей принадлежат одному классу, тем степень принадлежности этому классу для данного элемента выше). Затем для элемента, соответствующего тестовому изображению вычисляется степень его принадлежности к каждому из классов. При этих вычислениях используются степени принадлежности k ближайших соседей из обучающей выборки и расстояния до них (см., например, [13-14]). Эксперименты, проведённые с элементами нескольких известных баз изображений лиц, показали более высокий процент правильного распознавания нечёткой версии метода k ближайших соседей по сравнению с классической версией данного метода [14]. Отметим, что данный пример не исчерпывает всех разновидностей алгоритмов распознавания лиц, использующих элементы нечёткой логики.

Как в случае алгоритмов сегментации, где результатом работы алгоритма является отнесение каждого пикселя изображения к одному (в случае «чёткого» алгоритма) или нескольким (в «нечётком» случае) классам-сегментам, так и в случае алгоритмов распознавания лиц результатом является классификация (как правило однозначная) каждого изображения из тестового набора. Поэтому методика оценки алгоритмов, изложенная в предыдущем разделе, применима в обоих случаях.

Первый опыт применения данной методики к оценке алгоритмов распознавания лиц содержится в [6]. При этом были учтены упомянутые выше основные особенности таких алгоритмов. Для экспериментов нами была выбрана известная база изображений ORL. Она включает 400 образов, содержащих изображения лиц 40 людей (по 10 изображений каждого человека) [15]. Все изображения сделаны в полутоновом режиме, при незначительной вариации освещённости и отличаются выражением лица, поворотами головы и другими деталями.

Также для экспериментов нами был создан ряд новых изображений на основе образов базы ORL. Для этого использовалась свободно распространяемая программа морфинга изображений Sqirlz Morph версии 2.1. Она позволяет по двум изображениям лица осуществлять преобразование начального изображения лица в конечное изображение с сохранением промежуточных результатов.

В качестве тестируемых алгоритмов распознавания лиц в основном использовались демонстрационные версии программ, доступные на сайте http://www.advancedsourcecode.com/ посвящённом программному обеспечению в области компьютерного зрения и биометрии.

Тестирование проводилось с использованием упоминавшейся выше системы PICASSO.

Перед началом распознавания определялись нечёткие ground truth образы. Именно, каждому элементу *x* тестовой выборки *X* (куда могли входить и элементы обучающей выборки алгоритма) предписывались априори определенные степени принадлежности каждому из распознаваемых классов. Именно, для каждого $x \in X$ определялись $g^1_{C_1}(x),\ldots,g^1_{C_N}(x)$ так, чтобы $G =< X, g_1,\ldots, g_N >$ являлась нечёткой классификацией *X*. Все тестируемые алгоритмы выдавали однозначный результат распознавания. Соответственно, для выборки *X*, элементы её тестовой нечёткой классификации $F =< X, f_1,\ldots, f_N >$ определялись следующим образом: $f^2_{C_i}(x) = 1,$ если по результатам распознавания *x* отнесли к классу C_i и $f^2_{C_j}(x) = 0$ для *j* отличных от *i*.

По завершении тестирования вычислялись меры OA_1 и OA_2, определённые по формулам (11)-(12), а также доля правильно распознанных изображений (обычная статистическая мера). При этом был получен ряд содержательных результатов. Например, тестировалась устойчивость распознавания при небольших поворотах головы распознаваемого объекта. При сравнении между собой алгоритмов Eigenface и lbp общий процент правильно распознанных изображений был выше у второго алгоритма. При использовании нашей методики, изображениям поворотов головы априорно давалась более высокая степень принадлежности (равная 1), в сравнении с остальными изображениями (степень принадлежности равна 0.8). Соответственно, правильно распознанные повороты головы оцениваются выше, чем остальные правильно распознанные изображения. В результате, значения OA_1 и OA_2 для алгоритма Eigenface оказались выше, чем для lbp-алгоритма. Таким образом, несмотря на более низкую производительность первого из этих алгоритмов в сравнении со вторым, он оказался более устойчивым к распознаванию поворотов головы (подробнее см. [6]). По сходной схеме тестировалась способность алгоритмов распознавать изображения лица в очках и без них.

Для тестирования свойства разделённости элементов обучающей выборки в признаковом пространстве (важного для эффективности распознавания), в [6] при помощи программы Sqirlz Morph из пар изображений, представляющих разные классы, были созданы наборы гибридных изображений. Для каждого гибридного изображения естественно задавать его степени принадлежности обоим родительским классам: чем ближе изображение к одному из этих классов, тем степень принадлежности к этому классу выше, и наоборот, тем ниже степень принадлежности ко второму родительскому классу. Это даёт естественный способ создания нечётких ground truth образов для тестирования указанного свойства. Примеры родительских и гибридных изображений приведены на Рис. 1.

В качестве обучающей выборки было взято 30 изображений из ORL, представляющих 10 человек (по 3 обучающих образа на человека). Затем из некоторых пар обучающей выборки делались наборы гибридных изображений, из которых выбирались для тестирования по 72 изображений со степенями принадлежности родительским классам (0.9,0.1),...(0.55,0.45),(0.45,0.55),...,(0.1,0.9). Степень принадлежности остальным 8 классам для такого набора полагалась равной нулю.

Рисунок 1: Верхний ряд - исходные изображения из базы ORL, Нижний ряд - гибридные изображения. Степени принадлежности родительским классам для гибридных изображений, соответственно - (0.75,0.25), (0.5,0.5), (0.25,0.75).

В процессе тестирования вычислялись значения мер OA_1 и OA_2. В качества примера, количественные результаты тестирования нескольких алгоритмов на наборе гибридных изображений из Рис. 1 представлены в таблице 1.

Мера	Fisherface	Lbp	Eigenface
OA_1	0.5686	0.3582	0.5499
OA_2	0.725	0.5275	0.7151

Таблица 1. Результаты распознавания тестовых изображений, составленных из исходных изображений Рис. 1.

При непосредственной проверке оказалось, что алгоритм Fisherface отнёс каждое изображение из набора к тому родительскому классу, степень принадлежности к которому у изображения выше. Таким образом, разделение гибридного набора было выполнено корректно. Иная ситуация оказалась у lbp – алгоритма: из 72

изображений лишь 48 были отнесены к родительскому классу с преобладающей степенью принадлежности, 4 изображения – к родительскому классу с меньшей степенью принадлежности, а остальные 20 – к другим классам. У алгоритма Eigenface разделение гибридного набора было выполнено корректно, за исключением двух изображений (со степенями принадлежности (0.45,0.55) и (0.44,0.56)), отнесённых к другим классам. Таким образом, здесь значения указанных мер оказались адекватными реальной ситуации.

Вместе с тем, в [6] отмечалась желательность разработки новых нечётких мер сходства для более качественного тестирования. Так, одним из недостатков мер OA_1 и OA_2 являются относительно невысокие (т. е. далёкие от максимума, равного 1) их значения при успешной работе алгоритма (корректном или почти корректном разделении гибридного набора). Причина здесь в том, что нечёткие ground truth классы (со степенью принадлежности меньше 1) сравниваются с результатами распознавания, которые в нечёткой терминологии либо равны 1 при верной классификации, либо равны 0 в противном случае. Для преодоления данного недостатка, вместо OA_1 и OA_2 можно ввести однопараметрические семейства мер

$$OA_1INT^p\left(F^1,F^2\right)=\\ =\frac{\sum_i\sum_{x\in X}INT^p(f_{C_i}^1(x)\wedge f_{C_i}^2(x))}{\sum_i\sum_{x\in X}INT^p(f_{C_i}^1(x)\vee f_{C_i}^2(x))}, \quad (13)$$

$$OA_2INT^p\left(F^1,F^2\right)=\\ =\frac{\sum_i\sum_{x\in X}2INT^p(f_{C_i}^1(x)\wedge f_{C_i}^2(x))}{\sum_i\sum_{x\in X}INT^p(f_{C_i}^1(x)+f_{C_i}^2(x))}; \quad (14)$$

где *INT* - введённый Л. Заде оператор интенсификации контраста (contrast intensification operator) [16]:

$$INT(f(x))=\begin{cases}2f^2(x), & 0\le f(x)\le 0.5,\\ 1-2(1-f(x))^2, & 0\le f(x)\le 1,\end{cases}$$

а целочисленный параметр *p* - степень этого оператора (*p*=0,1,2...). При *p*=0 $OA_1INT^0=OA_1$, $OA_2INT^0=OA_2$. «Интенсификация контраста» состоит в том, что этот оператор увеличивает числовые значения большие 0.5 и уменьшает значения меньшие 0.5. Тем самым, функции принадлежности нечётких классов делаются ближе к характеристическим функциям обычных «чётких» множеств.

В терминах определённых таким образом мер при *p*=1 результаты, соответствующие приведённому выше примеру, представлены в таблице 2.

Мера	Fisherface	Lbp	Eigenface
OA_1INT^1	0.7053	0.4213	0.6812
OA_2INT^1	0.8272	0.5929	0.8037

Таблица 2. Результаты распознавания изображений Рисунка 1 в терминах мер (13)-(14) при *p*=1.

Сравнивая эти результаты с результатами таблицы 1, можно заметить, что увеличились как числовые значения мер, так и разности между элементами каждой строки. Последнее означает, что меры OA_1INT^1 и OA_2INT^1 более чувствительны к качеству классификации, чем меры OA_1 и OA_2.

Также, по аналогии с мерами (7)-(8) можно ввести меру *FFOA* (fuzzy figure of accuracy), сравнивающую тестовую $F=<X,f_1,...,f_N>$ и эталонную (ground truth) $G=<X,g_1,...,g_N>$ нечёткие классификации множества *X*:

$$FFOA(F,G)=\\ =\frac{\left[\frac{1}{M}\sum_{x\in X}\frac{1}{1+\sum_i|\alpha_i(f_i(x)-g_i(x))|^\beta}\right]-F_0}{1-F_0}, \quad (15)$$

где M – число элементов X, α_i, β – числовые (масштабные) параметры, F_0 – нормировочный коэффициент, представляющий оценку наихудшей классификации из возможных:

$$F_0=\frac{1}{M}\sum_{x\in X}\frac{1}{1+\sum_i|\alpha_i(f_i(x))|^\beta+|\alpha_i(g_i(x))|^\beta}.$$

В таблице 3 представлены результаты, соответствующие приведённому выше примеру, в терминах данной меры при значениях параметра β, равных 2 и 4 (параметры α_i во всех наших экспериментах полагались равными 1).

Как видно из этой таблицы, значения *FFOA* выше, чем соответствующие значения мер OA_1 и OA_2, при этом разности между элементами первой строки таблицы сопоставимы с соответствующими разностями таблицы 1. Разности между элементами второй строки

таблицы сопоставимы с соответствующими разностями таблицы 2. Это означает, что в данном примере меры OA_1INT^1, OA_2INT^1 и $FFOA$ при $\beta=4$ продемонстрировали примерно одинаковую чувствительность к качеству классификации.

Мера *FFOA*	Fisherface	Lbp	Eigenface
β=2	0.7748	0.5647	0.7605
β=4	0.9636	0.6938	0.9411

Таблица 3. Результаты распознавания изображений Рис. 1 в терминах меры (15) при различных значениях β.

В наших экспериментах ранжирование алгоритмов, сделанное на основе значений $FFOA$, не менялось при изменении значений параметра β. В то же время, при тестировании свойства разделённости обучающей выборки с использованием мер OA_1INT^p и OA_2INT^p, при разных значениях параметра p ситуация была иной. Так, для одного набора гибридных изображений, значения этих мер при p=1 для алгоритма Eigenface были выше, чем для алгоритма распознавания HOG, использующего гистограмму ориентированных градиентов изображения (см. [6]). Однако при p=2 ситуация поменялась на противоположную. Непосредственная проверка показала, что в случае алгоритма HOG, 4 изображения гибридного набора со степенями принадлежности (0.45,0.55),…,(0.42,0.58) были отнесены к родительскому классу с меньшей степенью принадлежности. В остальном распознавание было корректным (фактически, произошёл небольшой сдвиг в сторону одного из родительских классов). Что касается алгоритма Eigenface, то 2 гибридных изображения были отнесены алгоритмом к посторонним классам, а остальные изображения были классифицированы корректно. Таким образом, при различных значениях параметра p, эти ошибки разного типа имели разную цену.

4. Заключение

В рамках развиваемого нами подхода к сравнительному исследованию алгоритмов компьютерного зрения, разработка и анализ нечётких мер сходства (наряду с разработкой нечётких ground truth эталонов) играет важную роль.

Проведённые исследования показывают, что нечёткие меры сходства, применённые в сочетании со стандартными статистическими оценками производительности, позволяют получить дополнительную информацию о тестируемых алгоритмах.

Эксперименты с нечёткими мерами, используемыми при оценке алгоритмов классификации, позволили сформулировать основные требования к разрабатываемым мерам:

- Согласованность со здравым смыслом, т. е. удачным с точки зрения здравого смысла результатам работы алгоритма должны соответствовать высокие (обычно близкие к 1) значения мер.
- Высокая чувствительность значений мер к качеству классификации. При этом эти значения должны располагаться в широком диапазоне (обычно от 0 до 1).
- Простота численной реализации и понимание ситуаций, в которых применение мер оказывается полезным.

Последнее требование можно пояснить на приведённом выше примере сравнительного исследования двух алгоритмов распознавания лиц, включающем в себя сравнение их устойчивости к поворотам головы распознаваемого объекта.

В том примере статистические оценки производительности, дававшие преимущество одному алгоритму, дополнялись значениями мер OA_1 и OA_2, дававшими преимущество другому алгоритму.

Это позволило сделать вывод, что хотя общая производительность первого алгоритма выше, второй из них более устойчив к малым поворотам головы объекта.

Тем самым, именно различия между статистическими и нечёткими мерами сыграли здесь взаимно дополняющую роль. Использование мер OA_1INT^1 и OA_2INT^1 вместо OA_1 и OA_2 здесь нежелательно, так как легко видеть, что в этой ситуации первые из них ближе по поведению к статистическим мерам, чем вторые, и при их применении полезная дополнительная информация могла быть упущена.

В соответствии с последним сформулированным выше требованием, представляется полезным дальнейшее исследование поведения мер (13)-(15) в зависимости от их параметров.

Работа выполнена в рамках НИР № 0065-2014-0012 («Создание системы, позволяющей получить объективную оценку качества работы разнородных алгоритмов решения прикладных задач.»).

On the use of fuzzy similarity measures in evaluation of the computer vision algorithms.

A.S. Osipov

Abstract: In the paper some issues related to the performance evaluation of the computer vision algorithms within an empirical approach to their evaluation, developed by SRISA RAS, is considered. This approach uses elements of fuzzy set theory, in particular, fuzzy similarity measures. Some segmentation quality evaluation criteria are considered and their fuzzy extensions are offered. Some new fuzzy similarity measures applicable to the practical evaluation of face recognition algorithms, are introduced.

Keywords: performance evaluation, image segmentation, face recognition, ground truth images, fuzzy set theory, fuzzy similarity measures

Литература

1. H. R. Tizhoosh, Fuzzy Image Processing: Introduction in Theory and Applications, Springer-Verlag, 1997.

2**.** J.C.Bezdek, J.Keller**,** R.Krisnapuram**,** N.Pal. Fuzzy models and algorithms for pattern recognition and image processing. The handbooks of fuzzy set series. vol. 4, 2006, Springer Science & Business Media.

3. M. Banerjee, M.K. Kundu. Content Based Image Retrieval with Fuzzy Geometrical Features. Proceedings of the IEEE International Conference on Fuzzy Systems, 2003, pp. 932-937.

4. И.В. Грибков, А.В. Захаров, П.П. Кольцов, Н.В. Котович, А.А Кравченко, А.С. Куцаев, А.С. Осипов. Некоторые вопросы количественной оценки производительности детекторов границ. Программные продукты и системы, 2011, №4, стр.13-20.

5. А.В. Захаров, П.П. Кольцов, Н.В. Котович, А.А. Кравченко, А.С. Куцаев, А.С. Осипов, Критерии оценки качества сегментации изображений. Труды НИИСИ РАН, 2012, том 2, № 2, стр.87-99.

6. А.С. Осипов, Об использовании элементов нёчеткой логики в оценке алгоритмов идентификации лиц. Труды НИИСИ РАН, 2016, том 6, №2, стр 62-69.

7. A. Osipov. A fuzzy approach to performance evaluation of edge detectors. Lecture Notes in Signal Science, Internet and Education, 2007, WSEAS Press, pp. 94-99.

8. G. Jäger, U Benz. Measures of classification accuracy based on fuzzy similarity. IEEE Transactions on Geoscience and Remote Sensing, 2000, vol. 38, no. 3, pp. 1462-1467.

9. J.-H. Zhai, C.-Y. Bai, S.-F. Zhang. Face recognition based on 2DPCA and fuzzy -rough technique. Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC 2010), July 2010, pp. 725-729.

10. W.A.Yasnoff, J.K.Mui, J.W.Bacus, Error measures for scene segmentation, Pattern Recognition, 1977, vol.9, no.4, pp.217-231.

11. K.C.Strasters, J.J.Gerbrands, Three-dimensional image segmentation using a split, merge and group approach, *Pattern Recognition Letters*, 1991, vol.12, No.5, pp.307-325.

12. Y.J.Zhang, J.J.Gerbrands,, Objective and quantitative segmentation evaluation and comparison, Signal Processing, 1994, vol.39, No.1-2, pp.43-54.

13. J.-H. Zhai, C.-Y. Bai, S.-F. Zhang. Face recognition based on 2DPCA and fuzzy -rough technique. Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC 2010), July 2010, pp. 725-729.

14. X. Li. Face recognition method based on fuzzy 2DPCA. Journal of Electrical and Computer engineering, 2014, vol. 2014, Article ID 919041, 7 pages.

15. База ORL изображений лиц лаборатории AT&T. Доступна по адресу: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

16. L.Zadeh. Calculus of fuzzy restrictions. In Fuzzy Sets and Their Applications to Cognitive and Decision Processes, L. A. Zadeh, et aI., ed. 1975, New York: Academic Press, pp.1-39.

Особенности использования конфигурационных файлов при интеграции технологий параллельной обработки сигналов, приема данных по высокоскоростному каналу, подготовки запуска задач в мультипроцессорных комплексах реального времени

Т.К. Грингауз[1], А.Н. Онин [2]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН»,
Отдел математического обеспечения, Москва, Россия,
E-mail's: [1]gring@niisi.ras.ru, [2]alexii@niisi.ras.ru

Аннотация: Рассматриваются мультипроцессорные комплексы реального времени на базе процессоров КОМДИВ64-РИО, КОМДИВ128-РИО с коммуникационной средой RapidIO. В НИИСИ РАН разработана линейка программных изделий, включающая библиотеку параллельной обработки сигналов, пакет поддержки приема и передачи данных по высокоскоростному каналу, утилиты поддержки запуска программ. Разработка прикладных программ, предназначенных для функционирования под управлением операционной системы реального времени, связана с совместным использованием программных изделий. Применение каждого из последних требует создания и обработки своего набора конфигурационных файлов. Входная информация для разных программных изделий частично пересекается. На примере конкретных задач иллюстрируется технология совместного применения программных изделий, обеспечивающая минимизацию повторного ввода информации в конфигурационные файлы.

Ключевые слова: вычислительная стадия, группа процессоров, поток данных, высокоскоростной канал, конфигурация ВСК, коммуникационная среда RapidIO, статическая инициализация, оптимизация, распределение задач по процессорам, сценарий программы ПЗУ

1. Введение

Статья продолжает серию публикаций [6,7], посвященных технологии разработки программного обеспечения мультипроцессорных цифровых вычислительных комплексов с коммуникационной средой стандарта RapidIO[1] (далее - ЦВК) на основе аппаратных средств и средств общего программного обеспечения (далее - ОПО), разрабатываемых в ФГУ ФНЦ НИИСИ РАН. В состав ОПО входят библиотека параллельной обработки сигналов [2](далее - БПОС), пакет поддержки приема и передачи данных по высокоскоростному каналу [3] (далее – ПП ВСК-РИО), утилиты поддержки запуска программ [4] (далее – УПЗ-РИО). Все перечисленные программные изделия ориентированы на поддержку разработки унифицированного кода прикладных программ для функционирования на процессорах ЦВК под управлением операционной системы реального времени семейства ОС РВ Багет [5] (далее - ОС РВ). Под унификацией понимается разработка программ для многопроцессорной среды в таком виде, который позволяет загрузить на все процессоры один и тот же программный код. В этом случае выбор участка кода для исполнения осуществляется динамически по ключу, определяемому на основании уникального идентификатора процессорного элемента в среде RapidIO [1] (далее – РИО-идентификатор). РИО-идентификаторы присваиваются элементам среды RapidIO при ее инициализации [1]. РИО-идентификатор локального процессора определяется во время исполнения программы посредством специальной функции ОС РВ. Для настройки программы под конкретный процессор используются «конфигурационные данные», включаемые в текст программы (например, в виде заголовочных файлов). Этап составления

файлов с конфигурационными данными («конфигурационных файлов») технологически отделен от этапа разработки кода программ, что делает возможным масштабирование и настройку программного обеспечения под разные конфигурации аппаратных средств без изменения основного кода.

В статье описана технология использования конфигурационных файлов при интеграции трех программных изделий. Применение технологии проиллюстрировано двумя примерами.

2. Краткое описание технологий и принципов их интеграции. Основные понятия и определения

2.1 Интеграция БПОС и УПЗ-РИО

В [2] описана парадигма параллельного программирования и ее реализация в БПОС. В технологии БПОС задача обработки данных представляется в виде последовательности стадий вычислительного конвейера, принимающих и передающих потоки данных. Стадия может выполняться параллельно на нескольких процессорах, на одном процессоре могут выполняться несколько стадий. Процессоры разбиваются на группы так, что на всех процессорах группы выполняется один и тот же набор стадий. Количество процессоров в разных группах может отличаться. Каждому процессору соответствует «логический номер» - сквозной номер в упорядоченном множестве {{<номер группы>, <номер процессора в группе>}}. Логический номер в формализме БПОС используется для идентификации исполняемого на процессоре набора стадий. Масштабирование задачи осуществляется за счет использования «файла конфигурации задачи», в котором описываются основные примитивы БПОС (группы, стадии, потоки) с указанием количества процессоров в группе и типом разбиения потоков[2].

В [4] описаны назначение, функциональность, основные принципы применения УПЗ-РИО, а также технология интеграции БПОС и УПЗ-РИО. УПЗ-РИО выполняется на инструментальной ЭВМ под управлением ОС Linux с процессором архитектуры Intel x86/ x86_64. УПЗ-РИО обеспечивает привязку прикладной программы к конкретной конфигурации аппаратных средств и автоматизирует совокупность процедур для решения следующих задач:

- статическая инициализация среды RapidIO [1];
- оптимизация распределения вычислительных стадий задачи по процессорам на основе информации о потоках данных в системе с целью минимизации максимальной нагрузки на физическое соединение устройств RapidIO;
- создание сценариев программы ПЗУ для загрузки программного обеспечения в микропроцессоры (далее - «пользовательский сценарий»).

Автоматизация упомянутых процедур основана на использовании формализованного описания аппаратуры комплекса (далее - «файл аппаратной конфигурации») и потоков данных.

Привязка программы к аппаратуре осуществляется следующим образом. Разработчик должен программировать межпроцессорный обмен данными с использованием символических имен РИО-идентификаторов. Соответствие символических имен физическому расположению устройств устанавливается в файле аппаратной конфигурации вручную или автоматически (по умолчанию или в результате оптимизации). Соответствие числовых значений РИО-идентификаторов физическому расположению устройств устанавливается процедурой статической инициализации среды RapidIO, выполняемой программой ПЗУ по сценарию, порожденному УПЗ-РИО на основе файла аппаратной конфигурации. С помощью специальной опции
УПЗ-РИО можно сгенерировать файл с макроопределениями в формате языка Си, устанавливающий соответствие символических имен и числовых значений РИО-идентификаторов, которые будут присвоены устройствам по выполнении пользовательского сценария. Файл с макроопределениями включается в прикладную программу, что обеспечивает инвариантность основного текста относительно числовых значений РИО-идентификаторов.

Аппаратная конфигурация комплекса описывается в специфическом текстовом формате. Формат обеспечивает описание комплекса в виде совокупности именованных приборов, соединённых связями по протоколу RapidIO 1x/4x Serial. Прибор описывается как совокупность сборочных единиц (модулей) с привязкой к позициям на объединительной плате. Связи подсистем описываются посредством указания именованных разъёмов

соединяемых модулей. Поддерживается следующая номенклатура модулей и объединительных плат: ЦП-РИО-64А/Б/В/Г, ЦП-РИО-128А/Б/В, МР-РИО-А/Б/В, МП-ВПО-А/Б/В, М-РИО-А/Б, М-К128-А/Б; ОП-РИО-А/Б/В/Г, МП-ВПО. Формат позволяет указывать РИО-идентификаторы оконечных или гибридных устройств RapidIO [1] в составе модуля: СБИС 1890ВМ6Я [4] (КОМДИВ64-РИО), СБИС 1890ВМ7Я [4] (КОМДИВ128-РИО), СБИС 1890ВГ18Я [3] (краткое описание аппаратных средств приведено в [4]). РИО-идентификаторы можно указывать в числовом или символьном виде.

Оптимизация распределения задач по процессорам основана на информации о потоках данных в системе. В общем случае для описания потоков данных используется файл текстового формата «flow».

Для конфигурирования функциональности и определения параметров пользовательского сценария предназначен файл текстового формата «content».

УПЗ-РИО обладает опциональной возможностью обработки файла конфигурации задачи, реализованной по архитектурным правилам БПОС. В этом случае должен выполняться ряд правил, обеспечивающих интеграцию БПОС и УПЗ-РИО:

- в тексте программы символические имена РИО-идентификаторов процессоров, на которых должна выполняться задача БПОС, должны формироваться по шаблону: «<имя группы>_<номер процессора в группе>»,
- в файле конфигурации задачи должны быть указаны в виде специфических комментариев следующие данные:
  - количественные характеристики потоков (темп передачи данных в условных единицах),
  - «тип узла» для каждой группы процессоров. Примеры возможных значений типа узла: «K128» - процессоры КОМДИВ128-РИО; «K64» - процессоры КОМДИВ64-РИО; «K64|K128» - процессоры КОМДИВ64-РИО или КОМДИВ128-РИО; «K128<VSK» - процессоры КОМДИВ128-РИО, на которых планируется прием данных ВСК (пакетов типа NWRITE [1]).

В случае выполнения перечисленных выше условий символические имена РИО-идентификаторов процессоров, перечисленных в файле конфигурации задачи, можно не указывать в позициях РИО-идентификаторов файла аппаратной конфигурации. УПЗ-РИО автоматически формирует символические имена РИО-идентификаторов на основе файла конфигурации задачи и вносит их в файл макроопределений символических имен (далее - «файл ОСИ»). В этот же файл дополнительно автоматически вносятся определения следующих массивов:

- RIO_MAP - массив РИО-идентификаторов, упорядоченный по логическим номерам процессоров,
- RIOSTAT_VSK_RECEIVERS - массив РИО-идентификаторов процессоров, на которых планируется прием данных ВСК (пакетов типа NWRITE).

Массивы RIO_MAP, VSK_RECEIVERS используются в тексте программы для настройки ее под локальный процессор.

Массив RIO_MAP передается на вход функции mp_init(), входящей в состав ОС РВ. Функция возвращает РИО-идентификатор и логический номер локального процессора.

Массив RIOSTAT_VSK_RECEIVERS передается на вход функции vskSetProcessors(), содержащейся в программе «Библиотека ВСК» в составе ПП ВСК-РИО [3]. Функция передает операционной системе перечень РИО-идентификаторов процессоров, на которых планируется прием пакетов типа NWRITE.

В режиме интеграции с БПОС УПЗ-РИО устанавливает соответствие символических имен физическому расположению устройств и отображает его в файле аппаратной конфигурации автоматически (по умолчанию или в результате оптимизации, в зависимости от используемых опций). Если задача полностью описывается формализмом БПОС, то информация о потоках извлекается только из файла конфигурации задачи (в этом случае файл формата «flow» составлять не нужно).

2.2 Краткое описание ПП ВСК-РИО.

В [3] описаны архитектура, функциональность и принцип работы ПП ВСК-РИО, а также технология разработки прикладных программ с использованием ПП ВСК-РИО.

Высокоскоростной канал (ВСК) - это двунаправленный последовательный интерфейс «точка-точка», используемый в системах обработки цифровой информации для ввода данных от аналого-цифровых преобразователей (АЦП). Тракт данных от АЦП к процессорам КОМДИВ128-РИО включает специальные аппаратные устройства («устройства ВСК») на RapidIO: контроллер ВСК микросхемы 1890ВГ18Я и канал контроллера DMA для приёма данных ВСК через интерфейс RapidIO процессора КОМДИВ128-РИО. Краткая справка по архитектуре и функциональности устройств ВСК приведена в [3].

Данные ВСК поступают в контроллер ВСК по двум каналам (канал №1, канал №2). Каждый из каналов подразделяется на «канал А», «канал В» (далее – «порт А», «порт В»). Каждый из каналов 1А,1В, 2А, 2В может принимать входной поток данных ВСК или передавать поток данных в последовательный канал. Таким образом, контроллер может принимать до 4 входных потоков данных ВСК [3].

Контроллер ВСК имеет возможность перекрашивать входящие данные с целью дальнейшей идентификации данных от одной оптической линии. Всем данным, соответствующим одному входному потоку, приписывается определённый цвет (зеленый или красный) [3]. Цвет задаётся при конфигурировании контроллера. Далее применительно к одной микросхеме 1890ВГ18Я будем говорить о четырех потоках: («канал №1, красный», «канал №1, зеленый», «канал №2, красный», «канал №2, зеленый»).

Контроллер ВСК разбивает каждый из четырех потоков на сегменты и распределяет сегменты по процессорам через коммуникационную среду RapidIO (разные сегменты одного потока могут адресоваться разным процессорам). Для обеспечения требуемого распределения данных по процессорам необходимо предварительно соответствующим образом сконфигурировать микросхему контроллер ВСК.

Данные от контроллера ВСК поступают по RapidIO в принимающий процессор КОМДИВ128-РИО (далее - «принимающий процессор»). Копирование поступивших данных в память процессора осуществляется посредством КВСК[3]. Перед началом приема данных КВСК должен быть сконфигурирован программой, функционирующей на локальном процессоре. Детальное описание принципа работы КВСК приведено в [3].

КВСК может принимать данные ВСК от 32 источников (от восьми микросхем 1890ВГ18Я, до четырех потоков данных от каждой микросхемы). Источник идентифицируется совокупностью трех признаков: РИО-идентификатор микросхемы 1890ВГ18Я, номер канала (№1 или №2) и цвет данных. РИО-идентификаторы микросхем 1890ВГ18Я, от которых разрешен прием данных, указываются при конфигурировании ВСК.

Поддержка конфигурирования устройств ВСК, приёма и передачи данных в формате ВСК под управлением ОС РВ осуществляется посредством ПП ВСК-РИО. В состав ПП ВСК-РИО входят две программы: «Конфигуратор ВСК» и «Библиотека ВСК».

Конфигуратор ВСК – это инструментальное средство, предназначенное для составления конфигурационных данных устройств ВСК на основании информации о потоках данных ВСК в системе. Функционирует в среде Linux на инструментальной ЭВМ (далее – ИЭВМ) с процессором архитектуры Intel x86/ x86_64. Позволяет с помощью графического интерфейса задать распределение потоков входных данных по принимающим процессорным элементам на RapidIO. Создаёт соответствующий этому распределению «файл конфигурации ВСК», содержащий представление конфигурационных данных устройств ВСК в виде целочисленного массива в формате языка Си (далее – «массив конфигурационных данных»). В прикладной программе конфигурационные данные должны быть определены как целочисленный массив и инициализированы с помощью файла конфигурации ВСК. Файл конфигурации ВСК с массивом конфигурационных данных включается директивой #include в тексты прикладных программ, предназначенных для функционирования на процессорах КОМДИВ 64/128-РИО. Указатель на массив конфигурационных данных передается в качестве входного параметра функциям библиотеки ВСК.

Библиотека ВСК обеспечивает настройку устройств ВСК, приём и передачу данных в формате ВСК на аппаратных средствах комплекса под управлением ОС РВ.

2.2 Интеграция ПП ВСК-РИО с БПОС и УПЗ-РИО

Унификация текста программ для мультипроцессорной среды при использовании ПП ВСК-РИО обеспечивается следующим образом. Массив конфигурационных данных, включаемый в текст программы, содержит совокупность конфигурационных данных для всех ВСК-устройств, участвующих в приеме-передаче потоков ВСК. Формат массива конфигурационных данных обеспечивает возможность идентифицировать локальные и удаленные устройства ВСК, подлежащие конфигурированию с локального процессора, и вычленять конфигурационные данные для каждого из этих устройств. Идентификация устройств и вычленение конфигурационных данных производится функциями библиотеки ВСК. В качестве ключа, по которому производится идентификация процессора и относящихся к нему конфигурационных данных, используется РИО-идентификатор.

РИО-идентификаторы всех устройств ВСК, участвующих в конфигурировании или в приеме данных, передаются конфигуратору ВСК в составе входных данных. Конфигуратор позволяет использовать символические имена РИО-идентификаторов. Символические имена РИО-идентификаторов должны быть определены с помощью макросов #define в специальном заголовочном файле (далее - «файл списка идентификаторов»). В случае использования конфигуратором файла списка идентификаторов этот файл должен включаться в прикладную программу. Если подготовка запуска задачи осуществляется посредством УПЗ-РИО, то файл списка идентификаторов создавать не нужно – достаточно включить в прикладную программу файл макроопределений РИО-идентификаторов, сгенерированный УПЗ-РИО.

В случае использовании ПП ВСК-РИО совместно с БПОС и УПЗ-РИО необходимо соблюдать следующие правила.

1) При составлении файла конфигурации ВСК использовать символические имена РИО-идентификаторов микросхем 189ВГ18Я и принимающих процессоров.

2) При составлении файла конфигурации ВСК символические имена РИО-идентификаторов принимающих процессоров должны соответствовать файлу конфигурации задачи БПОС и формироваться по шаблону:

<имя группы>_<номер процессора в группе>.

3) В файле конфигурации задачи группе процессоров, выполняющих стадию приема данных от контроллеров ВСК, присвоить тип «K128<VSK».

4) В файле конфигурации задачи указать размер потоков (при необходимости оптимизации распределения задач по процессорам).

5) Составить файл описания потоков данных между процессорами и контроллерами ВСК в формате «flow» с указанием размера потоков (при необходимости оптимизации распределения задач по процессорам) и использовать этот файл наряду с файлом конфигурации задачи при выполнении оптимизации.

Примечание. Потоки данных ВСК не могут быть указаны в файле конфигурации задачи, поскольку не описываются средствами БПОС. В формализме БПОС описание потока определяет передачу данных между вычислительными стадиями. Поскольку микросхема 1890ВГ18Я не содержит процессорных элементов, на ней не могут функционировать вычислительные стадии, и ее невозможно указать в качестве источника или приемника потока в терминологии БПОС.

6) В файле аппаратной конфигурации указать символические имена РИО-идентификаторов микросхем 189ВГ18Я, использованные в файле конфигурации ВСК.

7) Если физическое местоположение микросхем 189ВГ18Я, принимающих или передающих данные ВСК, не подлежит оптимизации, необходимо пометить их символические имена в файле аппаратной конфигурации признаком «:».

Примечание. Местоположение микросхем 189ВГ18Я, используемых в режиме ВСК, нельзя выбирать произвольно. Такие микросхемы могут быть локализованы только на модулях, соединенных кабелем SRIO напрямую с источником внешних данных или с модулем МП-ВПО ([3,4])

8) При необходимости автоматического распределения задач по процессорам не указывать в файле аппаратной конфигурации символические имена процессоров, выполняющих вычислительные стадии БПОС.

9) В тексте прикладной программы на языке Си директивой #include включить следующие файлы:

- файл макроопределений РИО-идентификаторов, сгенерированный посредством УПЗ-РИО,
- файл конфигурации ВСК, сгенерированный конфигуратором ВСК.

10) В головной функции прикладной программы на языке Си должны быть определены массивы RIO_MAP, VSK_RECEIVERS и вызваны функции mp_init(), vskSetProcessors().

11) В тексте прикладной программы на языке Си конфигурационные данные ВСК должны быть определены как целочисленный массив и инициализированы с помощью файла конфигурации ВСК.

3. Примеры применения программных изделий

3.1 Общие положения

Ниже приведены два примера использования УПЗ-РИО для подготовки задач, реализованных по архитектурным правилам БПОС, к запуску на целевой ЭВМ (далее - «ЦЭВМ»). Схема ЦЭВМ приведена на рисунке 1 (темным оттенком выделены модули с интерфейсом RapidIO).

Первая задача полностью описывается формализмом БПОС, вторая – осуществляет передачу и прием данных ВСК и требует интеграции с ПП ВСК-РИО. Описание первого

примера содержит пошаговую инструкцию применения УПЗ-РИО, описание второго примера иллюстрирует только особенности, связанные с поддержкой потоков данных ВСК. Пошаговая инструкция приведена в методических целях, поскольку на практике соответствующая последовательность вызовов УПЗ-РИО и конфигуратора ВСК описывается в make-файле и выполняется путем однократного вызова команды make.

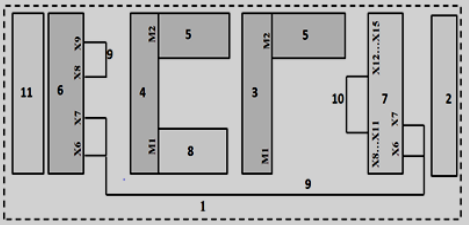


Схема целевой ЭВМ

Поз.	Наименование	Кол-во
1	Корпус с установленной кросс-платой ОП-РИО-В, ЮКСУ.468363.057	1 шт.
2	Модуль БТРС-011	1 шт.
3	Модуль ЦП-РИО-128А	1 шт.
4	Модуль ЦП-РИО-64А	1 шт.
5	Модуль М-К128-А	2 шт.
6	Модуль МР-РИО-А	1 шт.
7	Модуль МП-ВПО-А	1 шт.
8	Плата мезонинная БТМРС4-301А	1 шт.
9	Кабель SRIO	3 шт.
10	Кабель оптический LC-LC (спаренный)	8 шт.
11	Блок вентиляторов	1 шт.

Таблица 1. Состав целевой ЭВМ

Рисунок 1 Структурная схема и состав ЦЭВМ

Для работы использовался стенд, состоящий из ИЭВМ и ЦЭВМ, соединенных сетью Ethernet.

3.2 Пример 1 (интеграция БПОС и УПЗ-РИО)

3.2.1 Описание задачи

Задача реализует три стадии вычислительного конвейера:

- dout0 – стадия имитации данных и отправки их в выходной поток на стадию ретрансляции, для этого выделяется группа из двух процессоров с именем generator (размер передаваемой порции данных – 256 байт);

- dfw0 – стадия ретрансляции данных (приём данных из входного потока и отправка их в выходной поток на стадию приёма данных), для этого выделяется группа из двух процессоров с именем dfw;

- din0 – стадия приёма данных из входного потока, для этого выделяется группа из двух процессоров с именем receiver.

Примечание. При выполнении примера 1 разъемы X8,X9 модуля МР-РИО не были замкнуты кабелем SRIO.

3.2.1 Пошаговая инструкция по подготовке запуска задачи на целевой ЭВМ

3.2.1.1 Шаг 1. Организация структуры каталогов для ведения проекта.

В корневом каталоге проекта /project размещаются следующие папки:

-stend – папка для размещения конфигурационных файлов стендов в формате УПЗ-РИО (файл аппаратной конфигурации стенда и файл конфигурации пользовательского сценария). Если предполагается запуск задачи на разных конфигурациях аппаратуры, то для каждой из конфигураций в папке stend создаётся индивидуальная папка. Для примеров 1,2 использовалась папка «/713»;

- task – папка, в которой размещается программный код задачи, включая файл с головной функцией (main.c) и файл конфигурации задачи (taskConfig.c);

- tmp – папка для хранения временных файлов, например: файлы с TAR архивами образов ОС РВ (k128.tar, k64.tar), файл пользовательского сценария (userscript.psl), файл статической инициализации среды RapidIO (rapidio.bmrw), файлы протоколов утилиты rio-stat и др), target – папка для хранения целевых папок ОС РВ.

Структура каталогов для примера 1 (приведенное ниже содержимое папки stend соответствует состоянию по завершении шага 10 инструкции):

```
-/project
 -/tmp
  -/713
   …..
 -/stend
  -/713
              content
              hard.cfg
              hard.optimal

 -/forward
  -/task
              main.c
              taskConfig.c
              user_task.h
```

Далее будем считать, что вызовы исполняемого файла УПЗ-РИО rio-stat производятся в корневом каталоге /projects.

3.2.1.2 Шаг 2. Создание файла аппаратной конфигурации ЦЭВМ в формате УПЗ-РИО (файл project/stend/713/hard.cfg, рисунок 2).

В соответствии с приведенным описанием, на плате ОП-РИО-В установлены следующие модули: МР-РИО (позиция A01), ЦП-РИО-128 с мезонином М-К128 (позиция A02), ЦП-РИО-64 (позиция A03) с мезонинами БТМРС4-301 (электронный диск) и М-К128, МП-ВПО (позиция A02). Разъемы X6, X7 модуля МР-РИО соединены кабелями SRIO с одноименными разъемами модуля МП-ВПО.

3.2.1.3 Шаг 3. Редактирование файла конфигурации задачи (файл /project/forward/task/taskConfig.c).

В файле конфигурации задачи необходимо указать тип процессора для каждой группы и темп передачи данных для каждого потока. На рисунках 3,4 приведены фрагменты отредактированного файла конфигурации задачи (изменения выделены темным оттенком).

```
* OP-RIO-V 713
+ A01: MR-RIO
+ A02: CP-RIO-128+M2:M-K128
+ A03: CP-RIO-64A+M1:БТМРС4-301+M2:M-K128
+ A04: MP-VPO

* {*}-A01:X6 {*}-A04:X6
* {*}-A01:X7 {*}-A04:X7
```

Рисунок 2. Файл аппаратной конфигурации ЦЭВМ

3.2.1.4 Шаг 4. Оптимизация распределения стадий задачи по процессорам.

На основе конфигурационных файлов ЦЭВМ и задачи, составленных на шагах 2 и 3 соответственно, можно автоматически получить оптимизированное распределение стадий задачи по процессорам. Для этого необходимо на ИЭВМ выполнить следующую команду:

rio-stat -k <file2> -C <file3> -O "<mode>" < <file1>

с параметрами:

file1 - имя файла аппаратной конфигурации (получен на шаге 2), file2 - имя файла конфигурации задачи, функционирующей под управлением БПОС (получен на шаге 3), file3 - имя файла аппаратной конфигурации с расстановкой символических имен РИО-идентификаторов по результатам оптимизации (далее - «файл альтернативной аппаратной конфигурации»).

Параметр mode задаёт режим оптимизации распределения потоков данных, передаваемых между процессорами.

Например,

rio-stat -k forward/task/taskConfig.c -O "RHL"
-C stend/713/hard.optimal < stend/713/hard.cfg

```
#include <ps_din.h>
#include <ps_dout.h>
#include <ps_dfw.h>


#define GFLOW_SIZE  8

  pgroup_t pgroups[] = {
  /* Группа для генераторов */
    [0] = { .name = "generator",   .size = 2,
            /* .type = "K64|K128", */ },
  /* Группа для ретранслятора */
    [1] = { .name = "dfw",    .size = 2,
            /* .type = "K64|K128", */ },

    /* Группа для приемников */
    [2] = { .name = "receiver",    .size = 2,
            /* .type = "K64|K128", */ },
  };

 struct stage_info stages[] = {
  /* 0: генератор для входного потока*/
  [0] = {
    .name = "dout0",
    …
  .bind = { .group = &pgroups[0], },
  },

   /* 1: ретранслятор входного потока */
  [1] = {
    .name = "dfw0",
    …
    .bind = { .group = &pgroups[1], },
  },

  /* 1: приемник выходного потока 0 */
  [2] = {
    .name = "din0",
    …
    .bind = { .group = &pgroups[2], },
  },
};
```

Рисунок 3. Файл конфигурации задачи, фрагменты описания групп и стадий

```
struct flow_info flows[] = {

{ // Поток 'dout0' -> 'dfw0':  передаваемые данные потока №1
.name = "data1", // .size = "8x256x1x1", .type_sg = "NxN",
.src_stage = &stages[0],   .src_flow = 0,      .src_size = GFLOW_SIZE,
.dst_stage = &stages[1],   .dst_flow = 0,      .dst_size = GFLOW_SIZE,
.dst_sg = FLOW_SG_NO,
},
{ // Поток 'dfw0' -> 'din0':  передаваемые данные потока №1
.name = "data2", // .size = "8x256x1x1", .type_sg = "NxN",
.src_stage = &stages[1],   .src_flow = 0,      .src_size = GFLOW_SIZE,
.dst_stage = &stages[2],   .dst_flow = 0,      .dst_size = GFLOW_SIZE,
.dst_sg = FLOW_SG_NO,
},
};
```

Рисунок 4 Файл конфигурации задачи, описание потоков

Примечание. В режиме "RHL" минимизируются два критерия: максимальная нагрузка на физическое соединение и суммарная дальность пересылки («hop» [1]). Из двух расстановок лучшей считается та, которая улучшает значение одного из критериев, не ухудшая другого.

В результате выполнения команды файл альтернативной программной конфигурации

«hard.optimal» содержит протокол выполнения оптимизации. В протоколе в виде комментариев отображаются значения оптимизируемых критериев по ходу оптимизации и по ее результату. В конце протокола приводится финальная расстановка символических имен в файле аппаратной конфигурации. В рассматриваемом примере выводятся следующие сообщения о ходе оптимизации:

load 4096.00, hop 10

load 4096.00, hop 10

load 2048.00, hop 10

load 2048.00, hop 8

Оптимизация 1: максимальная загруженность канала - 2048.00, общее количество hop - 8

(видно, что улучшились значения обоих критериев). На рисунке 5 приведены для сравнения две расстановки символических имен: 1) в результате оптимизации, 2) по умолчанию.

```
С оптимизацией:
*OP-RIO-V   713
…
+A02:…,     receiver_0    generator_0  dfw_0
+A03:…,     dfw_1         generator_1  @     receiver_1
…

По умолчанию:
*OP-RIO-V   713
…
+A02: …     generator_0   generator_1  dfw_0
+A03:…,     dfw_1         receiver_0   @     receiver_1
…
```

Рисунок 5. Расстановки символических имен. Имена расставлены в соответствии с позициями вычислительных элементов на модуле

3.2.1.5 Шаг 5. Генерация заголовочного файла макроопределений символических имён (файл ОСИ).

На ИЭВМ выполнить следующую команду:

rio-stat -s <prefix> -d <file2>-k<file3> -I <path>[[:<file4>]...[:<fileN>]] < <file1>

с параметрами: file1 - имя файла аппаратной конфигурации прибора, file2 - имя выходного файла ОСИ, prefix - префикс, добавляемый к автоматически создаваемым символическим именам (назначается пользователем), path - путь, который будет использоваться для поиска файлов альтернативной аппаратной конфигурации, file4 … fileN - файлы альтернативной аппаратной конфигурации, file3 - имя файла конфигурации задачи (полученного на шаге 3).

В примере 1 команда выглядит так:

rio-stat -k forward/task/taskConfig.c -s SYS -d forward/task/user_task.h –I stend/713/hard.optimal < stend/713/hard.cfg

Фрагмент файла ОСИ usertask.h приведен на рисунке 6. К имени массива RIO_MAP добавлен префикс “ SYS ” в соответствии с выбранным значением опции –s.

3.2.1.6 Шаг 6. Получить графическое представление описания комплекса в формате «PNG» (этот шаг не является обязательным). Графическое представление описания прибора можно получить с помощью следующей последовательности команд:

rio-stat -g <file2> -k <file5> -I <path>[[:<file6>]...[:<fileN>]] < <file1>

dot -Tpng <file2> -o <file4>

с параметрами: file1 - имя файла аппаратной конфигурации прибора, file2 - имя выходного файла в формате DOT, file4 - имя выходного файла изображения описания прибора в формате PNG, file5 - имя файла конфигурации задачи (полученного на шаге 3), path – путь к файлам альтернативной аппаратной конфигурации, file6… fileN – файлы альтернативной аппаратной конфигурации.

В примере 1 команда выглядит так:

rio-stat -g tmp/713/shema.dot -k forward/task/taskConfig.c -I stend/713/hard.optimal < stend/713/hard.cfg

dot -Tpng tmp/713/shema.dot -o tmp/713/shema.png

Примечание. На рисунке не отображается модуль МП-ВПО. Этот модуль не требуется для выполнения задачи. Чтобы его отобразить, необходимо указать опцию «-X PIPE-SHOW=enable».

3.2.1.7 Шаг 7. Сборка образов ОС РВ.

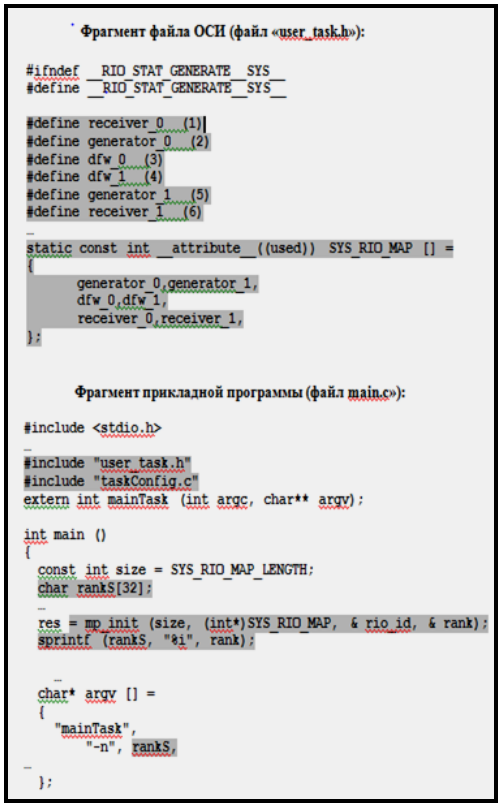
Образы ОС РВ собираются с прикладной задачей в соответствии с документацией на ОС РВ. Полученные файлы образов ОС РВ должны быть скопированы в выделенный каталог (tmp/713) для последующей автоматической генерации пользовательского сценария, предназначенного для загрузки файлов прикладной задачи на процессоры ЦЭВМ.

3.2.1.8 Шаг 8. Создание файла конфигурации пользовательского сценария (файл content).

Конфигурация пользовательского сценария составляется в специфическом текстовом формате «content», описанном в документации по УПЗ-РИО. Файл content должен быть размещен в каталоге проекта, созданной на шаге 1 (далее - «каталог генерации PSL-скриптов»). В самом файле должны быть определены:

- пути размещения файлов;

- пути для построения tar-файлов;
- определение tar-файлов (имя tar-файла и перечень файлов, которые должны быть помещены в архив);
- определение путей к файлам.



Фрагмент файла ОСИ (файл «user_task.h»):

```
#ifndef __RIO_STAT_GENERATE__SYS__
#define __RIO_STAT_GENERATE__SYS__

#define receiver_0   (1)
#define generator_0   (2)
#define dfw_0   (3)
#define dfw_1   (4)
#define generator_1   (5)
#define receiver_1   (6)

static const int __attribute__((used)) SYS_RIO_MAP [] =
{
      generator_0,generator_1,
      dfw_0,dfw_1,
      receiver_0,receiver_1,
};
```

Фрагмент прикладной программы (файл main.c»):

```
#include <stdio.h>

#include "user_task.h"
#include "taskConfig.c"
extern int mainTask (int argc, char** argv);

int main ()
{
  const int size = SYS_RIO_MAP_LENGTH;
  char rankS[32];

  res = mp_init (size, (int*)SYS_RIO_MAP, & rio_id, & rank);
  sprintf (rankS, "%i", rank);

  char* argv [] =
  {
    "mainTask",
        "-n", rankS,

  };
```

Рисунок 6. Фрагменты файлов usertask.h и main.c

Содержимое файла content для примера 1:

```
# Адрес размещения пользовательского сценария в ОЗУ
StartAddressPSL: 0xa1000000
# Адрес буфера в ОЗУ для копирования файлов
BufferAddress:  0xa4000000
# Пути размещения файлов
PathDestination:  tmp/713
# Путь для построения tar-файлов на ИЭВМ
TarPathSource:   tmp/713
# Определение tar-файлов
Tar: k64.tar  oc64
Tar: k128.tar  oc128
# Определение вспомогательных переменных окружения
Env: idsK64  $rio_ids_k64
Env: idsK128  $rio_ids_k128
Env: RunRamAddress  /dev/ram@0xa3
Env:
RunRamFile
${RunRamAddress}000200,0x3c00
# Определение путей к файлам
PathInst:  tftp://192.168.100.79/project
# Скрипт инициализации среды RapidIO
RapidIO: rapidio.bmrw
PathRam:  ${RunRamAddress}000000
# Копируемые файлы
Copy: k64.tar  $idsK64
Copy: k128.tar  $idsK128
# Запуск прикладного ПО
StartRunCommand:
 psl $RunRamFile  $RunRamAddress  1 boot file
Start: $idsK64 $idsK128
```

3.2.1.9 Шаг 9. Создание пользовательского сценария, tar-архивов образов и файла статической инициализации среды RapidIO.

Для этого необходимо на ИЭВМ выполнить следующую команду:

rio-stat -m <master> -U <fileIn>:<fileOut> -I <path>[[:<file3>]...[:<fileN>]] < <file1> > <file2>

с параметрами: file1 - имя файла аппаратной конфигурации прибора, file2 – имя выходного файла статической инициализации среды RapidIO, master - символическое имя «мастер-процессора», fileIn - имя входного файла описания пользовательского, fileOut - выходной файл, пользовательского сценария в формате языка PSL программы ПЗУ, path - путь к файлам альтернативной аппаратной конфигурации, file3…fileN - файлы альтернативной аппаратной конфигурации. Под «мастер-процессором» понимается процессор, выполняющий инициализацию среды RapidIO.

Для примера 1 команда выглядит следующим образом:

rio-stat -U stend/713/content:tmp/713/userscript.psl -m dfw_1 -I stend/713:hard.optimal < stend/713/hard.cfg > tmp/713/rapidio.bmrw

По выполнении команды файлы rapidio.bmrw, userscript.psl и tar-архивы образов ОСРВ будут сохранены в каталоге tmp/713.

3.2.1.10 Шаг 10 Запуск сценария

Ниже приведена команда запуска «пользовательского сценария» в программе PMON на мастер-процессоре (КОМДИВ128-РИО или КОМДИВ64-РИО):

psl <путь>/userscript.psl RUN

В примере 1 на мастер-процессоре (КОМДИВ64-РИО) выполнялась следующая последовательность команд:

ifconfig dc0 192.168.100.101
psl tftp://192.168.100.79/project/userscript.psl RUN

3.3 Пример 2 (интеграция ПП ВСК-РИО, БПОС и УПЗ-РИО)

3.3.1 Описание задачи

Задача реализует 5 вычислительных стадий:

- vsk-sender – стадия имитации данных ВСК и отправки их в микросхемы 1890ВГ18Я. Для этого выделяется группа из двух процессоров КОМДИВ128-РИО с именем imiter;

- vsk-receiver – стадия приёма потока данных ВСК, выполняется в группе receiver, состоящей из одного процессора КОМДИВ128-РИО. У стадии два выходных потока: принятые данные отправляются в стадию data-test-np, поток диагностических сообщений отправляется в стадию state-receiver;

- data-test-np – стадия пословной проверки данных ВСК, выполняется в группе vsk_test, состоящей из одного процессора КОМДИВ128-РИО. У стадии один выходной поток: принятые данные отправляются в стадию din;

- din – выполняется в группе vsk_test, состоящей из одного процессора КОМДИВ128-РИО;

-state-receiver – стадия приёма диагностических сообщений, выполняется в группе state, состоящей из одного процессора КОМДИВ64-РИО.

Стадия имитации данных ВСК имитирует четыре потока данных ВСК и передаёт их в две микросхемы 1890ВГ18Я. Потоки данных ВСК формируются в виде пачек размером 41476 слов.

Потоки 1 и 2 передаются соответственно в каналы 1А и 1В микросхемы 1890ВГ18Я с именем VSK_2. Потоки 3 и 4 передаются соответственно в каналы 2А и 2В микросхемы 1890ВГ18Я с именем VSK_4.

Потоки ВСК, вышедшие из каналов 1А, 1В микросхемы VSK_2 и из каналов 2А, 2В микросхемы VSK_4 попадают соответственно в каналы 1А, 1В микросхемы VSK_1 и в каналы 2А, 2В микросхемы VSK_3. Далее все потоки передаются на принимающий процессор, выполняющий стадию vsk-receiver.

Примечание. Для реализации такой схемы передачи на рассматриваемой ЦЭВМ необходимо в качестве микросхем VSK1, …,VSK4 выбрать четыре микросхемы в составе модуля МР-РИО (раземы X6, X7, X8, X9). Передача данных в формате ВСК между VSK2, VSK1 будет осуществляться по оптическим линиям связи через модуль МП-ВПО, а между VSK4, VSK3 – по линии «витая пара» (кабвль SRIO). Предварительно кабелем SRIO должны быть замкнуты разъемы X8, X9 модуля МР-РИО

Стадия vsk-receiver собирает по одной пачке от каждого потока и формирует выходной «куб» размером [128 * 384 * 4] слов. Выходной куб отправляется на стадию data-test-np. На стадию state-receiver отправляется информация о текущем состоянии четырёх приёмных каналов – «куб» данных размером [4 * 1 * 1] слов.

Стадия vsk-receiver принимает куб и сравнивает его содержимое с эталоном. Затем куб данных передаётся в стадию din.

Стадия state-receiver принимает информацию о состоянии каналов, анализирует её и в случае обнаружения ошибок в каналах выводит информацию в консоль.

Стадия din принимает куб данных и поглощает его.

3.3.2 Особенности подготовки запуска задачи, связанные с поддержкой потоков данных ВСК

В задачах, связанных с приемом или передачей данных в формате ВСК, в дополнение к пошаговой инструкции, приведенной в разделе 3.2.1, необходимо выполнить еще ряд действий. Проиллюстрируем это с помощью примера 3.

3.3.2.1 В файле аппаратной конфигурации изначально расставить символические имена РИО-идентификаторов микросхем 1890 ВГ18Я (рисунок 7).

Файл аппаратной конфигурации ЦЭВМ

```
* OP-RIO-V 713
+ A01: MR-RIO          :VSK_1  :VSK_2  :VSK_3  :VSK_4
+ A02: CP-RIO-128+M2:M-K128
+ A03: CP-RIO-64A+M1:БТМРС4-301+M2:M-K128 :!master1=1
+ A04: MP-VPO
* {*}-A01:X6 {*}-A04:X6
* {*}-A01:X7 {*}-A04:X7
* {*}-A04:X8..X11 {*}-A04:X12..X15
```

Рисунок 7. Файл аппаратной конфигурации для примера 2. Кабель SRIO, соединяющий разъемы X8,X9 модуля МР-РИО, не указан в конфигурации, поскольку не используется для передачи пакетов SRIO (по нему передаются данные в формате ВСК).

3.3.2.2 Создать файл конфигурации ВСК с использованием символических имен (test-2.vsk) (рисунок 8).

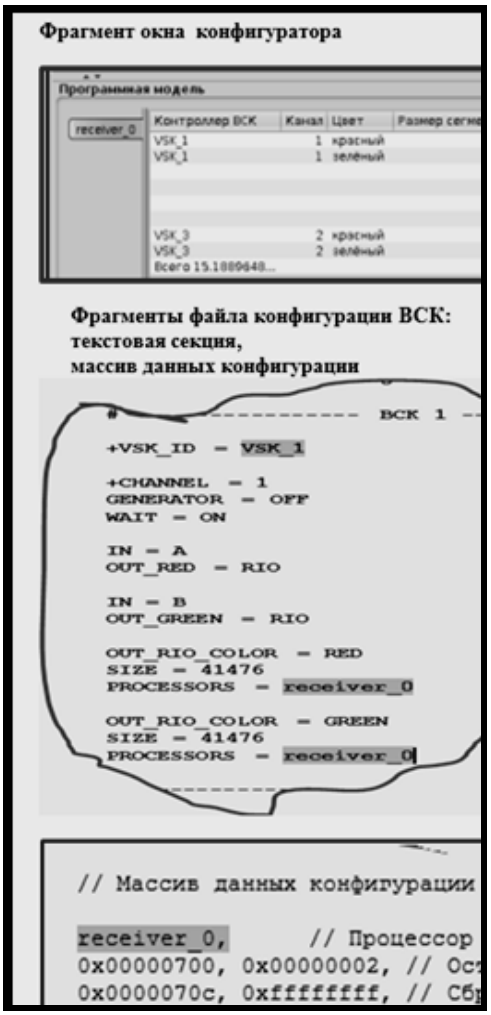


Рисунок 8.Фрагменты окна конфигуратора ВСК и файла конфигурации ВСК

3.3.2.3 В файле конфигурации задачи указать тип ВСК-приемника и размеры потоков (рисунок 9).

```
// Описание групп процессоров
pgroup_t pgroups[] =
{
  [0] = {.name = "imiter", .size = IMITER_COUNT,
                                  /* .type = "K128 ", */},
  [1] = {.name = "receiver", .size = RECEIVER_COUNT,
                          /* .type = "K128<VSK", */},
  [2] = {.name = "state",  .size = 1,
                                  /* .type = "K64", */},
  [3] = {.name = "vsk_test", .size = TEST_COUNT,
                          /* .type = "K128", */},
};
// Описание потоков
flow_info_t flows[] =
{
  ...
  {// Поток 'vsk-receiver' -> 'data-test-np':  принятые дан
   .name = "vsk_data", // .size = "4*128*384*4", .type_sg =
```

Рисунок 9 Фрагмент файла конфигурации задачи с указанием типа ВСК-приемника и размера потоков

3.3.2.4 Составить файл описания потоков ВСК в формате «flow», описанном в документации по УПЗ-РИО.

Для примера 2 был составлен файл task15.flow следующего вида:

VSK_1,VSK_2 - receiver_0 663616

imiter_0,imiter_1 - VSK_2,VSK_4 1327232

```
…
#include <user_task.h>
#include <etalons.h>

#include <ps_vsk_imit.h>
#include <ps_vsk.h>
#include <ps_vsk_state.h>
#include <ps_test_np.h>
#include <ps_din.h>
…
// Параметры запуска задачи ПОС
#define MAIN_TASK mainTask
#define MAIN_VSK_RECEIVERS
RIOSTAT_VSK_RECEIVERS
#define MAIN_RIO_MAP N_RIO_MAP
#define MAIN_RIO_MAP_LENGTH  N_RIO_MAP_LENGTH
…

#include <stdio.h>
#include <vsk.h>
…

// Функция запуска задачи ПОС
int  main (void)
{
  vskSetProcessors  (MAIN_VSK_RECEIVERS);

  int id;
  int rank = -1;
  const int size = MAIN_RIO_MAP_LENGTH;

  mp_init (size, (int*) MAIN_RIO_MAP,  &id,
&rank);

size_t nflows = sizeof  (flows) / sizeof
(*flows);
  char sizeS [32];
  sprintf  (sizeS, "%i", size);

  char rankS [32];
  sprintf  (rankS, "%i", rank);

  char* argv [] =
  {
    "mainTask",
    …

    "-n", rankS,
    "-s", sizeS,
    …
  };
```

Рисунок 10. Фрагменты файла, содержащего функцию запуска задачи для примера 2

В каждой строке файла перечислены РИО-идентификаторы источника (источников) и приемника (приемников) потока, а также его размер в условных единицах. Необходимо, чтобы при описании потоков в файле конфигурации задачи и в файле формата «flow» использовались одни и те же условные единицы (в данном случае – количество байт за итерацию).

3.3.2.5 Генерировать файл альтернативной аппаратной конфигурации с учетом файла описания потоков ВСК. Файл формата «flow» подключается в командной строке посредством опции «-c», например:

rio-stat -k task.c -C hard.optimal -O "HLR" –c "task15.flow" < hard.cfg. В файле ОСИ “user_task.h” для примера 2 будут присутствовать следующие определения массивов RIO_MAP, RIOSTAT_VSK_ RECEIVERS:

```
static const int __attribute__((used)) N_RIO_MAP [] =
{ imiter_0, imiter_1, receiver_0, state_0, vsk_test_0,};
…
static const unsigned int __attribute__((used)) RIOSTAT_VSK_RECEIVERS [] =
{receiver_0, 0};
…
```

3.3.2.6 Включить в файл, содержащий функцию main() файлы usertask.h, описать и инициировать RIO_MAP, RIOSTAT_VSK_RECEIVERS, массив конфигурационных данных ВСК, вызвать mp_init(), vskSet() (Фрагменты файла, содержащего функцию запуска задачи для примера 2, приведены на рисунке 10)

3.3.2.7. Включить в текст программы описание и определение массива конфигурационных данных, инициализированного посредством файла конфигурации ВСК test-2.vsk, например:

```
int configVSK_test2 [];
int configVSK_test2 [] =
{
#include "vsk/test-2.vsk"
};
```

Aspects of configuration files usage for integration of parallel digital signal processing, data reception via high-speed serial channel, software start in multiprocessor real-time systems technologies

T.K. Gringauz, A.N. Onin

Abstract: Multiprocessor real-time systems with RapidIO communications based on KOMDIV64-RIO, KOMDIV128-RIO microprocessors are considered. Digital signal processing system software includes parallel digital signal processing library and two development tool kits: one for high-speed channel devices programming and the other one for software start in multiprocessor RapidIO environment support. Development of applied programs destined to real-time operating system requires combined usage of all the three said software components. Each component requires specific set of configuration files, though informational content of different sets is partly shared. This paper presents technology of configuration files usage. Technology provides base procedures automation and minimization of content duplication in different files. Technology is illustrated using concrete examples.

Keywords: computing pipeline stage, processor group, data flow, high-speed channel, configuration data array, RapidIO communication environment, static initialization, optimization, tasks processor distribution, embedded script.

Литература

1. RapidIO Interconnect Specification (Revision 1.3) Available from: http://www.rapidio.org/specs/current.
2. Г.О. Райко. Библиотека параллельной обработки сигналов. // Труды НИИСИ РАН.- М., НИИСИ РАН, 2015, том 5, № 1, с. 64-69
3. Т.К. Грингауз, А.Н. Онин. Программное обеспечение для приема и передачи данных по высокоскоростному каналу в мультипроцессорных комплексах реального времени. // Труды НИИСИ РАН.-М., НИИСИ РАН, 2014, том 4, № 2, с. 22-32
4. Т.К. Грингауз, А.Н. Онин. Инструментальное программное обеспечение для подготовки запуска задач в мультипроцессорных комплексах реального времени. // Труды НИИСИ РАН.-
М., НИИСИ РАН, 2015, том 5, №2, с.122-129
5. А.Н. Годунов [и др.], Операционная система реального времени Багет 3.0.// Программные продукты и системы. - Тверь, Научно-исследовательский институт «Центрпрограммсистем», 2010, № 4, с. 15 – 19

Библиотека мониторинга для многопоточных программ

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия,.
E-mail's: [1]grntl@niisi.ras.ru, [2] kostas@niisi.ras.ru, [3] ashch@niisi.ras.ru

Аннотация: Статья посвящена вопросам реализации средств и механизмов контролируемого выполнения многопоточной прикладной программы в среде операционной системы реального времени, функционирующей на многопроцессорной вычислительной системе. Рассматриваются архитектура библиотеки мониторинга, технологические и архитектурные аспекты ее применения.

Ключевые слова: контролируемое выполнение, библиотека мониторинга, многопоточная программа, поток управления, очередь сообщений, сигнал, операционная система реального времени, многопроцессорная система.

Введение

В статье излагается назначение, структура и принципы реализации многопоточных прикладных программ (далее, приложений). Многопоточные программы, реализованные по принципу функциональной декомпозиции задачи, характеризуются модульностью и масштабируемостью [1]. Функционально независимые (или слабо зависимые) друг от друга части программы, реализованные с использованием отдельных потоков управления, могут свободно исключаться, заменяться и дополняться без внесения существенных (а зачастую и вообще каких-либо) изменений в другие части программы. Многопоточная программа более понятна, структурирована и, как следствие, более надежна по сравнению с программами, в которых средства декомпозиции не применяются.

При проектировании приложения в многопроцессорных системах сначала выполняется развёртывание программного обеспечения на процессорных модулях в соответствии с требованиями к аппаратуре, быстродействию, конвейеризации и объему передаваемых между процессорными модулями данных. В результате развёртывания получается «топология задачи» (термин введен по аналогии с «топологией сети») – схема расположения и взаимодействия компонентов программного обеспечения в многопроцессорной системе.

На следующем этапе проектирования выполняется функциональная декомпозиция программного обеспечения на каждом из процессорных модулей (т. е. на узлах в смысле топологии) многопроцессорной системы. В соответствии с [2] результатом декомпозиции, в типовом случае, должно быть выделение в отдельные группы потоков управления, наряду с фрагментами, реализующими основную функциональность приложения, следующих функциональных фрагментов программного обеспечения:

– фрагментов, обеспечивающих асинхронный ввод-вывод; программирование конкретных особенностей ввода-вывода может выполняться в специализированных потоках управления, что снижает логическую сложность программы;

– фрагментов, обеспечивающих диалог с оператором;

– фрагментов, обеспечивающих связь с другими программами, например, графический сервер, геоинформационная система, база данных.

Библиотека мониторинга позволяет реализовать функции, которыми должна обладать система, спроектированная в соответствии с принципами контролируемого выполнения [2, 3, 4]. В соответствии с [2], программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы в критических условиях (например, при поступлении в программу некорректных данных, ошибок в среде исполнения или в самой прикладной программе).

Средства контролируемого выполнения прикладной программы содержат механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, а также генерацию управляющих воздействий, в случае отклонения поведения программы от эталона.

Реализация средств контролируемого выполнения требует наличия в прикладной программе агентов, аккумулирующих данные о выполнении (контрольные параметры состояния), монитора, анализирующего в реальном масштабе времени получаемые от агентов контрольные параметры состояния и генерирующего управляющие воздействия (команды управления приложением), а также средств выполнения сгенерированных монитором команд управления приложением. Кроме того, в прикладной программе должны присутствовать средства передачи контрольных параметров состояния от агентов к монитору и средства, передающие команды управления прикладной программой.

1. Архитектура библиотеки мониторинга

Архитектура библиотеки мониторинга, однопроцессорной системы и структура прикладной программы, работающей совместно с библиотекой мониторинга на одном процессоре, рассмотрена в [2].

Структура прикладной программы, работающей совместно с библиотекой мониторинга на многопроцессорной системе, представлена на рисунке 1.

Рисунок 1. Библиотека мониторинга в многопроцессорной системе

При работе библиотеки мониторинга на многопроцессорной системе используются удалённый (главный) и локальный (подчиненный) мониторы. Главный монитор назначается при конфигурировании библиотеки мониторинга, при этом прочие мониторы считаются подчиненными.

В многопроцессорной системе соответствие между ГФП прикладной программы к процессоруосуществляется посредством идентификаторов коммуникационной среды RapidIO (идентификаторы процессоров назначаются при конфигурировании многопроцессорной системы).

Связь между главным и локальными мониторами выполняется с использование эмуляции Ethernetего эмуляции в коммуникационной среде RapidIO. Запись протокола работы (лога) главного монитора выполняется по протоколу **nfs.**

Для организации связи удалённого монитора с главным монитором используется дополнительные потоки управления, конвертирующиепоступающие от главного потока локального монитора сообщения, приходящие через очередь сообщений, в пакеты, передаваемые по Ethernet. Соответственно, данные, поступающие от главного монитора, перенаправляютсячерез очередь сообщений главному потоку локального монитора. На рисунке 1поток управления **EthSrv** используется для приема сообщений от

главного монитора, а потоки **EthCli1** выполняет передачу сообщений главному монитору.

2. Технологические аспекты применения библиотеки мониторинга

Применение библиотеки мониторинга предполагает использование следующей технологии проектирования. На первом этапе определяется структура многопоточного приложения: количество групп потоков управления, количество потоков управления в группе, выполняет распределение групп потоков управления по процессорным элементам многопроцессорной системы. Затем спроектированная структура многопоточного приложения переносится в файл конфигурации БМ.

Перенос проекта в файл конфигурации БМ может быть выполнен двумя способами. Первый способ состоит в ручном порождении это файла. Файл конфигурации БМ – это информационный модуль на языке Си, содержащий определения типов данных и переменных, также в этом файле выполняется инициализация заданных переменных. Таким образом, данных способ конфигурирования состоит в написании кода на Си, компиляции и сборки.

При втором способе конфигурирования используются графические средства программирования. Пользователь (разработчик системы) заполняется в ходе интерактивного сеанса значения конфигурационных полей. В редакторе многопоточного приложения программист, используя визуальные средства, составляет структуру приложения и задает свойства его объектов (групп потоков управления, непосредственно потоков управления).

При втором способе конфигурирования используетсянезависимо компилируемый компонент (плагин) MTA-application для Eclipse. Плагин предоставляет средства верификации структуры приложения и генерации конфигурации БМ в виде объектного файла, пригодного для подключения к ОСРВ.

Программист должен вручную заполнить некоторые свойства объектов многопоточного приложения при составлении конфигурации в плагине MTA-application для Eclipse.

Для объекта многопроцессорных систем должны быть указаны значения следующих атрибутов:

– имя многопроцессорной системы

– идентификатор многопроцессорной системы

– сетевой порт главного монитора многопроцессорной системы

– старшие 3 октета сети многопроцессорной системы с точками;

– идентификатор коммуникационной среды RapidIO главного монитора;

– многопроцессорной системы;

– количество процессоров в многопроцессорной системе.

Для объекта «Поток управления» должны быть заполнены следующие свойства:

имя пользовательской функции (например, «**osUserThread**»);

– **third_iters** – количество итерации в цикле потока управления

Прочие свойства объектов «Многопроцессорная система», «Группа потоков управления» и «Поток управления» инициализируются значениями «по умолчанию», однако при необходимости могут быть изменены программистом.

3. Архитектурные аспекты применения библиотеки мониторинга

Библиотека мониторинга не только порождает свою инфраструктуру, но и формирует определённую архитектуру приложения. При использовании библиотеки мониторинга часть ресурсов прикладной программы создаётся автоматически на основе данных, содержащихся в конфигурационном файле. К таким ресурсам относятся потоки управления, семафоры и очереди сообщений. Количество потоков управления, семафоров и очередей сообщений указывается при создании конфигурационного файла. Соответствующие ресурсы генерируются автоматически.

В принципе, запрет на динамическое порождение ресурсов отсутствует, но корректность исполнения такой программы не гарантируется средствами библиотеки мониторинга. Более того, соответствие требованиям библиотеки мониторинга может быть проверено средствами статического анализа. Если, например, вызов функций **pthread_create( )** в тексте приложения отсутствует, то считается, что БМ сконфигурирована корректно и используется в соответствии с правилами.

В библиотеки мониторинга реализована концепция порождения потоков управления при старте – все потоки управления в заданной группе на заданном процессорном элементе порождаются при старте системы. Таким образом, роль головной функции приложения берёт на себя библиотека мониторинга.

В связи этим в прикладной программе отсутствуют в явном виде последовательное порождение потоков управления. Последовательное порождение потоков удобно тем, что некоторый поток управления может инициализировать ресурсы и проконтролировать корректность их инициализации, а затем породить дочерние потоки, которые эти ресурсы будут использовать. Приложение, использующее БМ, должно быть спроектировано таким образом, чтобы связанные потоки управления (например, в части использования ранее порожденных ресурсов) имели явную синхронизацию. Например, если один поток инициализирует ресурсы, и другие потоки используют эти ресурсы, то требуется синхронизировать заданные потоки через контролируемый библиотекой мониторинга семафор (или множество семафоров) таким образом, чтобы потоки, использующие ресурсы, ожидали семафор, устанавливаемый потоком, который инициализирует ресурсы.

4. Особенности проектирования и разработки прикладных программ, работающих совместно с БМ

Для проектирования и разработки прикладных программ, предназначенных для выполнения в многопроцессорных системах совместно с библиотекой мониторинга, может быть использована среда разработки Eclipse с предустановленным плагином ТСАГ СПО [5].

Плагин ТСАГ СПО предоставляет прикладному программисту широкий набор средств для автоматизированного проектирования и разработки прикладных программ, предназначенных для выполнения в многопроцессорных системах совместно с библиотекой мониторинга. Использование этих средств сокращает трудоемкость интеграции библиотеки мониторинга и готовых (написанных ранее) прикладных программ, уменьшает время, затрачиваемое на документирование и конфигурирование прикладных программ с явной многопоточностью для многопроцессорных систем.

Плагин ТСАГ СПО позволяет в автоматизированном режиме создавать модели многопроцессорной системы в визуальном редакторе, заполнять свойства многопроцессорной системы в диалоговом режиме и генерировать файл конфигурации.

Плагин ТСАГ СПО для среды разработки Eclipse позволяет использовать все функциональные преимущества данной среды (например, редакторы файлов с исходным текстом на языке Си, XML и XSD редакторы) при разработке проектов ТСАГ СПО, а также предоставляет прикладному программисту гибкие и настраиваемые средства для компиляции и сборки программных модулей.

Проектирование и разработка задачи в ТСАГ СПО выполняется в следующей последовательности:

– создание модели многопроцессорной системы;

– заполнение свойств объектов модели многопроцессорной системы;

– синхронизация с файловой системой – создание системы каталогов, соответствующей объектам модели многопроцессорной системы;

– создание диаграмм программных модулей объектов модели многопроцессорной системы;

– генерация программных модулей объектов модели многопроцессорной системы;

– создание модели многопоточного приложения;

– связывание объектов модели многопроцессорной системы и объектов модели многопоточного приложения;

– компиляция проекта модели многопроцессорной системы;

– компиляция проекта модели многопоточного приложения;

– загрузка прикладной программы на целевой ЭВМ.

Важной особенностью ТСАГ СПО является наличие встроенных средств коллективной разработки. Эти средства предоставляют прикладному программисту возможность ведения разработки проектов ТСАГ СПО на удаленном сервере. Доступ к удаленной файловой системе организован с помощью RSE (RemoteSystemExplorer). Работа с удаленным сервером выполняется по защищенному каналу, реализованному с помощью протокола SSH.

Использование RSE является прозрачным для прикладного программиста

– удаленный проект выглядит в Eclipse так же, как и локальный. Следует отметить, что использование механизмов удаленной работы с проектами позволяет организовать разработку в режиме ограниченного доступа, когда на локальном компьютере у разработчика нет прав на сохранение данных. В этом случае, он работает с удаленным проектом без ограничений, однако не может сохранить данные на локальный компьютер. Разработка на удаленном сервере может выполняться из сетей общего пользования – уровень безопасности обеспечивается безопасностью протокола SSH.

Плагин ТСАГ СПО совместим со встроенными в Eclipse средствами версионированиягit, например, EGit или JGit.

5. Выводы

Приведённая в статье архитектура прикладного программного обеспечения позволяет использовать библиотеку мониторинга для создания приложений, соответствующих принципу контролируемого выполнения.

Разработана технология, обеспечивающая автоматизацию создания программ, использующих библиотеку мониторинга. Применение технологии приводит к созданию структурированных программ, гарантирующих управляемое использование примитивов операционной системы в составе приложения. Технология освобождает разработчика от рутинных действий по использованию типовых конструкций операционной системы.

В статье описывается подход к интеграции библиотеки мониторинга и разработанного ранее технологического пакета ТСАГ СПО. Изложенный в статье подход применим как к однопроцессорным, так и к многопроцессорным системам. Создание библиотеки мониторинга выполнялось в рамках работ ФГУ ФНЦ НИИСИ РАН по технологии разработки систем реального времени.

Monitor Library for Multithread Programs

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: Controlled execution implementation and mechanisms of an application running in a real-time operating system environment on a multiprocessor computer system is under consideration in the paper. Special attention is paid to monitor library architecture, and technological aspects of its application.

Keywords: controlled execution, monitor library, multithread program, thread, message queue, signal,real-time operating system, multiprocessing system.

Литература

1. В.Л.Безруков, А.Н.Годунов, П.Е.Назаров, В.А.Солдатов, И.И.Хоменков. Введение в ос2000, Вопросы кибернетики. Информационная безопасность, Операционные системы реального времени, Базы данных /Под ред. чл.-корр. РАН В.Б.Бетелина. - Москва; НИИСИ РАН, 1999, – с. 76-106.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН, Том 5 (2015), № 2, с. 113-121.
3. В.А. Галатенко. Контролируемое выполнение / В.А. Галатенко, К.А.Костюхин, Н.В.Шмырев – М: НИИСИ РАН, 2012. – 157 с.
4. В.Б.Бетелин. Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А.Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, № 6, с. 45-55.
5. Генератор текста программ в исходном виде для систем реального времени / К.Г. Нархов // Программные продукты и системы. – 2010, № 4, с. 24–30.

Встроенная система контроля

А.В.Науменков[1], С.А.Сидоров[2]

[1] *ОАО КБ «Корунд-М», Москва, Россия, e-mail: andrey_naumenkov@korund-m.ru*
[2] *ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН», Москва, Россия, e-mail: sidorov@niisi.msk.ru*

Аннотация: Описана *Встроенная система контроля* (ВСК) для многопроцессорных ЭВМ, решаемые ею задачи, принципы работы и особенности применительно к ЭВМ с различной коммуникационной средой.

Ключевые слова: тестирование, многопроцессорные ЭВМ, распределенная система

Введение

В ряду изделий разработки ФГУ ФНЦ НИИСИ РАН и ОАО КБ «Корунд-М» значительное место занимают ЭВМ различного назначения и конфигурации, построенные по модульному принципу [1]. Предыдущее поколение машин, ныне успешно применяемое во множестве систем, было построено на шине VME [2]. Сейчас им на смену пришли ЭВМ на базе коммутируемых каналов RapidIO [3]. Параллельно развивается линейка малогабаритных ЭВМ в конструктиве PC-104plus.

Большинство этих ЭВМ многопроцессорные с количеством процессоров до нескольких десятков. Кроме того, в машинах устанавливаются разнообразные беспроцессорные и мезонинные интерфейсные модули, в состав ЭВМ часто входят и периферийные устройства.

Важнейший вопрос, возникающий и при производстве, и при эксплуатации ЭВМ, – проверка работоспособности аппаратуры [4]. Для этого необходимы специальные программные средства тестирования.

Исторически сложились два взаимно дополняющих друг друга подхода к тестированию многопроцессорных ЭВМ: низкоуровневое тестирование автономной тестовой системой и тестирование в среде операционной системы (ОС). Каждый подход имеет как преимущества, так и ограничения.

В качестве низкоуровневого средства применяется Встроенная система контроля (ВСК), которой посвящена данная статья. Отсутствие ограничений ОС по настройке и использованию аппаратуры позволяет ВСК протестировать все режимы работы аппаратуры, все линии связи и провести достаточно тонкую диагностику, что как правило невозможно при работе через драйверы ОС. В то же время, отсутствие поддержки параллельных процессов делает весьма затруднительным, например, нагрузочное тестирование, и приводит к необ

ходимости последовательного выполнения тестов.

ОС как среда исполнения оказывается необходимой, прежде всего, на этапе разработки ЭВМ. ОС позволяет включить в работу одновременно множество устройств, создавая таким образом режим максимальной нагрузки на машину. Практически все стандартные наборы тестов для оценки производительности рассчитаны на ОС (обычно Linux). Также в рамках ОС функционируют и контрольные задачи пользователей, для которых ЭВМ разрабатывается [5].

В ФГУ ФНЦ НИИСИ РАН разработана и применяется тестовая система Пульсар для POSIX-совместимой ОС РВ Багет, предназначенная для комплексной проверки ЭВМ. К сожалению, работа эта не опубликована.

Пульсар представляет собой распределенную систему, части которой работают на каждом процессорном модуле. Взаимодействие осуществляется по стандартным коммуникационным интерфейсам, поддерживаемым операционной системой и представленным драйверами, в частности, Ethernet. При этом физическая среда передачи данных может быть различной, например, VME, реальный Ethernet, RS-232 или RapidIO. Программа тестирования определяется конфигурацией, задаваемой обычным текстовым файлом в простом формате. Основной задачей при разработке Пульсара было обеспечить комплексность тестирования, т.е. взаимодействие как можно большего числа одновременно работающих устройств и узлов ЭВМ, а также механизмов самой ОС, и эта цель была достигнута.

Тем не менее, для нужд производства и ремонта требовался иной подход, позволяющий проверять аппаратуру как таковую, локализовывать и диагностировать неисправности. Разработка такой тестовой системы низкого уровня была начата в КБ «Корунд-М» в 2000-м году и далее поддержана специалистами НИИСИ РАН, прежде всего в части наполнения тестами. Система получила название

Встроенная система контроля – ВСК или ETS – Embedded Test System.

Основные цели при разработке ВСК были такими:

- создать компактную программную тестовую систему с возможностью конфигурирования в терминах физических устройств прямо на целевой машине;

- обеспечить требуемые при производстве, ремонте и эксплуатации режимы проверок;

- обеспечить проверку внешних интерфейсов как наиболее уязвимых частей ЭВМ.

Термин «встроенная» появился от одного из основных режимов работы ВСК при эксплуатации ЭВМ - тестирование по включению питания, когда требуется постоянное присутствие ВСК в пользовательской флеш-памяти и запуск из Программы ПЗУ.

Режимы тестирования ВСК

На сегодняшний день ВСК функционирует на всех типах ЭВМ, разработанных обеими организациями, и является основным производственным инструментом проверки работоспособности аппаратной части машин. Определился и стабилизировался круг решаемых ею задач. Это:

Исследовательское тестирование. Разработка новой ЭВМ предполагает и создание новых тестов для нее, и одновременную отладку аппаратуры самой ЭВМ. Таким образом, на выходе появляется очередная версия ВСК с новыми тестами и, как правило, новыми элементами конфигурирования, а также отлаженная машина. Эта исследовательская версия служит базой для решения всех остальных задач ВСК.

Тестирование ЭВМ по включению питания. Этот вид тестирования, называемый часто начальным или POST – Power On Self Test, предполагает, что за весьма ограниченное время, как правило не превышающее двух минут, нужно проверить всю аппаратуру ЭВМ. Очевидно, что в полном объеме такую проверку выполнить невозможно, поэтому начальное тестирование – всегда компромисс между широтой охвата и глубиной тестирования. Дополнительные ограничения обусловлены необходимостью выполнения проверок в составе объекта эксплуатации, т.е. выход на внешние интерфейсы ЭВМ запрещен. В такой ситуации главным объектом тестирования становятся внутренние линии связи ЭВМ – каналы межмодульного взаимодействия и сопряжения с устройствами. Оставшееся время отводится на проверку доступности основных узлов и модулей ЭВМ, а затем и их функциональности.

Настройка начального теста выполняется всегда статически, ВСК не предусматривает никакого автоматизма или динамики в этом вопросе.

Важнейшим фактором начального тестирования является полное восстановление настроек аппаратуры по окончании тестирования, в том числе и в случае обнаружения неисправностей, поскольку следом за начальным тестом должна начать работу штатная программа, для которой необходимо обеспечить предсказуемую среду старта.

Результат начального тестирования должен быть доступен штатной программе в виде, по крайней мере, “OK/ERROR”, а если имеется возможность, то и в более развернутом формате. Решение по результатам тестирования принимает не тест, а штатная программа.

Регламентный контроль подразумевает полную проверку всей машины с учетом некоторых ограничений. Выполняется регламентный контроль в составе объекта эксплуатации, предназначен для углубленной проверки аппаратуры и определения неисправных элементов. При этом тесты осуществляют полную проверку соответствующих узлов и механизмов каждого модуля, входящих в состав ЭВМ устройств и внутренних коммуникаций.

Хотя проверка проводится в составе объекта эксплуатации, но выполняется в так называемом технологическом режиме, который не предполагает последующего выполнения штатной программы. На тест регламентного контроля не накладывается ограничений по времени, поэтому он может длиться от десятков минут до нескольких часов. Естественно, выход на внешние интерфейсы ЭВМ запрещен.

Тест регламентного контроля запускается на выполнение оператором командой с консоли, и результаты также выводятся на консоль в текстовом виде.

Диагностика неисправностей проводится с точностью до сменного модуля ЭВМ или канала связи. Основная задача этого теста – помочь персоналу на месте эксплуатации найти неисправный аппаратный компонент и восстановить работоспособность ЭВМ заменой модулей из ЗИП.

Начальное и регламентное тестирование относятся к эксплуатационным тестам. Далее описаны технологические режимы тестирования, применяемые на производстве.

Циклическое тестирование по глубине и охвату аналогично регламентному контролю, однако предназначено для проведение специальных видов испытаний, таких, как проверка на длительную непрерывную работу, темпера-

турные и прочие климатические испытания. ВСК бесконечно выполняет циклический тест, после каждого прохода на консоль выдается сводная информация об обнаруженных неисправностях или об отсутствии таковых.

Циклический тест предназначен для выполнения вне объекта эксплуатации, однако также не предусматривает выход на внешние интерфейсы.

Причина этого иная: специальные испытания проводятся как правило в жестких условиях, на которые рассчитана ЭВМ, но не дополнительная аппаратура, необходимая для проверки внешних интерфейсов.

Уровень диагностики в циклическом тесте максимальный, так как в условиях производства необходима точная локализация неисправностей, вплоть до компонентов электронных модулей.

Тест Технических условий – тест ТУ, наиболее полный и всеобъемлющий вид тестирования, поддерживаемый ВСК.

Как следует из названия, тест предназначен для проверки выполнения требований Технических условий на ЭВМ и применяется при проведении приемо-сдаточных испытаний. Он же может быть передан потребителям для проведения входного контроля приобретенных ЭВМ.

Тест ТУ выполняет полное тестирование аппаратуры с входом на внешние интерфейсы ЭВМ.

Это подразумевает необходимость использования внешних замыкателей и присоединяемых устройств, а также в отдельных случаях участие оператора в процессе тестирования, например, для контроля изображения на экране монитора и подтверждения его правильности нажатием клавиш на клавиатуре консоли.

Результат тестирования выводится на экран консоли, здесь важно итоговое сообщение типа “OK/ERROR”, хотя в большинстве случаев процесс сопровождается выдачей полного протокола проверки, где вначале сообщается наименование проверяемого узла, а затем результат.

Диагностическое тестирование предназначено для исследования и локализации неисправности, ранее обнаруженной при тестировании в других режимах.

Применяется преимущественно при ремонте и наладке оборудования.

Этот вид тестирования подразумевает углубленное тестирование отдельных подсистем – группы модулей или одного модуля, возможно его части по выбору оператора, а также выбор тестов для проверки.

Процесс тестирования

Многопроцессорные ЭВМ строятся по модульному принципу. Каждый процессорный модуль включает один или несколько микропроцессоров, причем каждый из их может содержать несколько процессорных ядер. Таким образом, понятие «многопроцессорная ЭВМ» может быть отнесено и к одному процессорному модулю. Поэтому для описания работы ВСК удобнее оперировать понятием «процессорный элемент» (ПЭ), означающим микропроцессорное ядро со своей памятью (программ и данных) и выходом на системообразующий канал связи ЭВМ, в нашем случае VME или RapidIO.

Конструктивно ЭВМ выполняется как корпус (крейт) с объединительной платой (кросс-платой), в которую вставляются электронные модули. Кросс-плата обеспечивает коммуникационные каналы для межмодульного взаимодействия и вывод некоторых интерфейсов модуля на внешние соединители ЭВМ.

ВСК представляет собой распределенную систему. Ее копии (экземпляры) работают одновременно на всех процессорных элементах под управлением выделенного главного (консольного) ПЭ. Обычно главным бывает ПЭ, к которому подключена консоль оператора. В задачу консольного экземпляра ВСК входит организация синхронного старта ВСК на всех ПЭ, сбор результатов тестирования и передача их, если нужно, программе штатного режима.

Работа каждого экземпляра ВСК определяется конфигурацией, заданной при настройке системы. В VME-машинах каждый модуль хранит только локальную конфигурацию, а консольный – еще и шкалу наличия модулей. В RapidIO-машинах в каждой копии ВСК имеется полное описание конфигурации ЭВМ, а нужный фрагмент для ПЭ определяется по физическому местоположению ПЭ в составе ЭВМ – географическому адресу. Требование предусмотреть возможность однозначного определения географического адреса всегда закладывается при проектировании ЭВМ. Поскольку в состав ЭВМ входят и пассивные (беспроцессорные) компоненты, такие, как мезонинные интерфейсные контроллеры и модули памяти, периферийные устройства (мыши, мониторы и пр.), а также полноразмерные пассивные модули (коммутаторы и пр.), проверка их функционирования возлагается на ближайшие процессорные элементы.

Состав проверяемой аппаратуры полностью определяется конфигурацией, а глубина тестирования – режимом. При этом тестирование выполняется до первой ошибки: при обнаружении неисправности работа ВСК прекра-

щается (за исключением циклического режима).

Подготовленная для исполнения ВСК оформлена как TAR-архив, в составе которого находится сам исполняемый двоичный образ ВСК и файлы описания конфигурации. Поскольку в составе ЭВМ могут быть микропроцессоры различных наименований, то исполняемый образ готовится для каждого наименования свой, при этом файлы конфигурации остаются одинаковыми. Этот TAR-архив перед началом тестирования должен быть размещен в пользовательской флеш-памяти каждого ПЭ, для чего в VME-машинах загрузка и запись ВСК выполняется при настройке ЭВМ вручную оператором, а в RapidIO-машинах, где количество ПЭ составляет десятки, для этой цели разработана специальная автоматизированная процедура, использующая возможности Программы ПЗУ.

Конфигурирование

Конфигурирование ВСК, т.е. описание состава и структуры тестируемой аппаратуры, может быть выполнено как на самой проверяемой ЭВМ, так и на инструментальной машине. Результатом конфигурирования является файл описания конфигурации, используемый затем тестовой системой для выбора выполняемых тестов.

Конфигурация ЭВМ состоит из двух частей:

- глобальная конфигурация (только для ЭВМ RapidIO);
- локальная конфигурация (для всех ЭВМ).

ЭВМ с шинами VME и PCI

В ЭВМ с шинами VME и PCI присутствует только локальная конфигурация. Все модули на этих шинах однопроцессорные. Локальная конфигурация включает все узлы модуля и присоединенную к нему аппаратуру (мезонины, мыши, клавиатуры, мониторы, модули VME), до которых может через свое адресное пространство дотянуться процессор модуля.

В ЭВМ с шиной VME в крейт может быть установлено несколько модулей, связанных между собой шиной VME. Каждый из модулей "видит" сигналы шины VME. При этом каждый модуль "знает" свое местоположение в крейте – он получает ее с кросс-платы крейта VME.

Модули, присутствующие в крейте, задаются при конфигурировании ВСК маской - шестнадцатеричным числом, единица в соответствующем разряде которого означает присутствие в конфигурации модуля с данным номером.

Один из модулей VME назначается консольным (с помощью программы ПЗУ). Консольный модуль проверяет сам себя и управляет тестированием прочих модулей. Консольный модуль не знает локальных конфигураций других модулей, а знает только об их наличии из маски. Таким образом, каждый модуль проверяет себя сам, а консольный модуль только выдает команду на начало проверки подчиненного модуля, отображает ее прохождение на собственной консоли и обрабатывает результат проверки. В результате, все операции по проверке ЭВМ и можно производить с консольного модуля.

ЭВМ с коммутируемыми каналами RapidIO

Коммуникационная среда RapidIO имеет более сложную топологию, чем VME. Это обусловлено технологией "точка-точка", применяемой в RapidIO. Основой топологии являются коммутаторы RapidIO, имеющие по нескольку портов, каждый из которых может быть связан с одним из портов соседнего коммутатора.

ЭВМ на базе RapidIO состоят из сегментов, каждый из которых состоит из 5 модулей. Сегмент имеет жесткую топологию - модули сегмента связаны между собой соединениями параллельного RapidIO (PRIO) по принципу звезды - каждый с каждым (правда, не все модули из пяти могут присутствовать). Внутри модуля к его основному коммутатору присоединены процессоры и другие коммутаторы. Модуль может содержать несколько процессоров. Из сегмента наружу выходят соединения последовательного RapidIO (SRIO). Способ расстановки соединений SRIO между сегментами не определен - это может быть звезда, кольцо или какая-либо другая топология.

Таким образом, ЭВМ RapidIO имеют топологию, которую невозможно описать одним 16-ричным числом, как линейную топологию VME. Поэтому топология RapidIO в ВСК описывается с помощью графа. Граф представляется списками вершин и соединений RapidIO.

В начальный момент, после включения ЭВМ, среда RapidIO находится в неинициализированном "разомкнутом состоянии". Существует специальный порядок инициализации среды RapidIO, описанный в ее спецификациях. Этот порядок характерен для сред с соединениями типа "точка-точка" и предполагает последовательную инициализацию всей среды из какой-либо одной исходной точки. В нашем случае такой точкой является процессор, на котором ВСК была запущена оператором (или скриптом).Инициализация среды RapidIO заключается в:

- присвоении всем конечным точкам (процессорам и прочим узлам, которые могут быть источниками и потребителями информации) уникальных идентификаторов - ID;

- заполнении таблиц маршрутизации коммутаторов среды, обеспечивающих доставку пакетов от одной конечной точки к другой.

Алгоритмы инициализации среды RapidIO могут быть разнообразными. В ВСК применен алгоритм, описанный в следующем разделе.

Этот простой алгоритм обеспечивает маршрутизацию пакета от любой конечной точки до любой другой конечной точки. Однако он может оставить незадействованными некоторые соединения RapidIO. Для проверки таких соединений производится перепрограммирование маршрутов непосредственно в ходе тестирования данных соединений.

Так как в одном модуле ЭВМ RapidIO может быть несколько процессоров, единицей управления и тестирования является уже не модуль, как в ЭВМ VME, а процессорный элемент, описываемый локальной конфигурацией. Адрес процессорного элемента обозначается при конфигурировании в виде "A(X:Y:Z)", где X- номер сегмента, Y - номер модуля, Z - номер ПЭ внутри модуля.

ЭВМ RapidIO в целом описывается глобальной конфигурацией, представляющей собой список модулей в привязке к сегментам и список соединений SRIO. Глобальная конфигурация должна быть задана правильно, в противном случае среда RapidIO не сможет быть проинициализирована. В ходе глобального конфигурирования предоставляется возможность задания локальных конфигураций процессорных элементов консольного и подчиненных модулей. Таким образом, полная конфигурация ЭВМ формируется на консольном модуле. В ходе инициализации RapidIO из списка модулей и соединений SRIO глобальной конфигурации генерируется граф, используемый далее при всех операциях ВСК со средой RapidIO.

Инициализация среды RapidIO

Инициализация среды RapidIO заключается в записи таблиц маршрутизации во все коммутаторы ЭВМ.

Для того, чтобы произвести запись информации с консольного процессора в коммутатор необходимо:

- знать расстояние (количество промежуточных коммутаторов) от консольного процессора до программируемого коммутатора и номер находящейся за ним конечной точки;

- предварительно запрограммировать таблицы маршрутизации всех промежуточных коммутаторов.

При запуске ВСК производится расчет последовательности команд инициализация RapidIO по следующему алгоритму:

1. По данным конфигурации ЭВМ, представляющей собой список модулей и соединений Serial RapidIO (SRIO), формируется граф системы, состоящий из таблицы вершин (микросхем коммутаторов и конечных точек) и таблицы соединений между ними (физических соединений Parallel RapidIO – PRIO и SRIO).

2. Рассчитываются таблицы маршрутизации коммутаторов:

- производится перебор всех вершин графа – конечных точек. Для каждой конечной точки:

- производится виртуальный обход графа в ширину. При этом при попадании в очередную вершину-коммутатор фиксируется порт, через который мы в него попали. Этот порт в связке с номером конечной точки будет впоследствии записан в список связок маршрутизации данного коммутатора.

3. Вычисляется информация, необходимая для доступа к конкретному коммутатору с консольного модуля (для того, чтобы задать с консольного модуля его таблицу маршрутизации).

Для каждого коммутатора это делается так:

- производится перебор всех вершин-конечных точек, кроме консольной. Для каждой из них:

- производится виртуальный проход графа, начиная от консольной точки, до заданной, с использованием уже вычисленных таблиц маршрутизации. При этом на каждом шаге маршрута контролируется, в какой мы попали коммутатор и сколько сделали шагов. Если нужный нам коммутатор обнаружен, заносим пару чисел {расстояние:конечная точка} в структуру, описывающую коммутатор. Если мы наткнулись на коммутатор второй раз, проверяем, меньше ли полученное расстояние, чем в уже имеющейся паре. Если меньше, меняем пару в структуре коммутатора.

Для системы в целом запоминается максимальная дальность коммутатора (MAXL).

Таким образом, в результате выполнения пунктов 1, 2 и 3 для каждого коммутатора среды RapidIO в памяти консольного процессора сформированы пара {расстояние:конечная точка} и таблица маршрутизации.

4. Далее производится непосредственная инициализация аппаратуры среды RapidIO. Это происходит следующим образом:

- производится перебор всех коммутаторов, находящихся на расстоянии 1 от консольного процессора. В каждый коммутатор записывается полученный ранее его список связок (п.2). Для адресации коммутатора использует-

ся пара {расстояние:конечная точка}, полученные в п. 3;

- то же делается для всех коммутаторов с дальностью 2;

- то же делается для всех коммутаторов с дальностью 3;

...

- то же делается для всех коммутаторов с дальностью MAXL.

Тиражирование и запуск ВСК

ЭВМ на RapidIO содержит до нескольких десятков процессорных элементов, поэтому технология машин VME с последовательной ручной записью ВСК во флеш-память каждого из подчиненных процессоров неприменима. Вместо этого используется процедура тиражирования ВСК. Процедура тиражирования запускается после окончания конфигурирования.

Для начала работы ВСК на ЭВМ с архитектурой RapidIO необходимо:

- записать TAR-файл ВСК во флеш-память какого-либо из ПЭ ЭВМ (далее будем называть его консольным);

- запустить ВСК в режиме конфигурирования и выполнить глобальное и локальное конфигурирование ЭВМ в целом.

Если конфигурирование уже было выполнено ранее, то:

- выполнить инициализацию среды RapidIO;

- выйти из ВСК и запустить скрипт тиражирования ВСК.

После этого необходимо перезапустить ЭВМ и можно исполнять ВСК в тестовых режимах.

Запустить ВСК на ЭВМ RapidIO можно только один раз. Для повторного запуска ВСК ЭВМ необходимо перезапустить. Это связано с тем, что в начальный момент работы ВСК использует контроллеры сообщений процессоров и протоколы передачи сообщений Программы ПЗУ для раздачи команд на запуск образов ВСК на подчиненных процессорах (другого способа запустить их не существует). В ходе проведения проверок среды RapidIO ВСК перепрограммирует контроллеры сообщений процессоров RapidIO, после чего восстановить исходное состояние контроллеров невозможно. Поэтому, для приведения системы в исходное состояние, необходимо ее перезапустить.

Особенности локального тестирования

Локальное тестирование представляет собой проверку ресурсов, до которых может дотянуться ПЭ через свое адресное пространство, то есть локализованное в пределах модуля. При этом максимальную трудность представляют проверки:

- «прозрачных» для обычного программиста устройств – кэш-памяти и TLB в составе процессорного ядра;

- основной памяти – из-за задействования ее программами ПЗУ и ВСК как системного ресурса;

- декодирования адресов различными контроллерами – из-за необходимости перенастройки адресных пространств контроллеров, используемых как системные ресурсы;

- дискретных сигналов ввода-вывода – из-за разнообразия их реализации и не определенной заранее схемы их внешнего межсоединения для проведения проверки;

- контроллеров DMA – из-за необходимости проверки выставления ими различных диапазонов адресов и весьма сложным устройством, например, контроллера СР2 процессора 1890ВМ7Я;

- интерфейсных устройств со сложными протоколами – SCSI, МКИО и т.д.;

- видеоконтроллеров – из-за большого количества поддерживаемых видеорежимов;

- устройств ввода-вывода информации – клавиатуры, мыши и мониторов.

Тестирование коммуникационных сред

Как шина VME, так и среда RapidIO представляют собой достаточно сложные многокомпонентные устройства, в значительной степени определяющие работоспособность ЭВМ в целом и, в то же время, наиболее подверженные внешним воздействиям в силу большого числа контактов в соединителях.

При тестировании шины VME проверяются следующие ее ресурсы:

- механизм декодирования адресов, сосредоточенный в контроллере шины;

- работа шин данных и адреса по передаче информации, включая перебор возможных форматов данных и конфигураций адресных пространств;

- генерация, передача и обработка сигналов прерывания;

- генерация, передача и обработка почтовых прерываний;

- работа со спинами.

ВСК проверяет все подчиненные модули попарно с консольным модулем. Особенностью тестирования является необходимость параллельного исполнения программы на двух модулях.

На ЭВМ RapidIOпроводятся следующие тесты:

- проверка географических адресов всех элементов;
- проверка технологических буферов памяти подчиненных процессоров с использованием только циклов maintenance;
- проверка режимов функционирования процессорных модулей, задаваемых внешними дискретными сигналами;
- проверка установки и наблюдаемости сигналов состояния ЭВМ, доступных всем ПЭ;
- проверка наличия линков там, где они должны быть в соответствии с таблицей конфигурации;
- проверка регистров и таблиц маршрутизации микросхем коммутаторов;
- проверка работы очередей и почтовых ящиков контроллеров сообщений процессоров;
- посегментная проверка передачи сообщений (последовательный перебор коротких маршрутов внутри сегмента и смежных с ним SRIO). Контроллер сообщений передает информацию в автоматическом режиме. Кроме стандартной схемы инициализации RapidIO при необходимости используются временные маршруты с последующим возвратом к стандартной схеме;
- межсегментная проверка, система рассматривается в целом и возможны маршруты до 12 элементов;
- проверка работы контроллеров doorbell процессоров;
- проверка правильности коммутации пакетов внутри коммутатора, т. е. прохождения пакетов от любого заданного порта к другим портам в соответствии с таблицей маршрутизации;
- тест максимальной нагрузки сети RapidIO. ПЭ начинают передавать информацию одновременно, поочередно отправляя пакеты другим процессорам, участвующим в тесте. Длина пакета выбрана минимальной (1024 байт) для обеспечения максимального количества переключений коммутации. Второй вариант теста максимальной нагрузки использует передачу информации контроллерами сообщений процессоров в автоматическом режиме по всем возможным портам, длина пакета 4096 байт.

Заключение

Встроенная система контроля продолжает развиваться и наполняться новыми тестами и механизмами проверки аппаратуры. Заложенные при ее создании технические решения, такие, как конфигурируемость на целевой ЭВМ, близость к нижнему аппаратному уровню и пр. остаются актуальными. На сегодняшний день ВСК является одним из ключевых инструментов проверки выпускаемого оборудования.

По мнению производственников, ВСК позволяет детектировать около 90% отказов аппаратуры и почти все грубые отказы – замыкания, обрывы и т.д.

Embedded Test System

A.V.Naumenkov, S.A.Sidorov

Abstract: Described Embedded Test System (ETS) for multiprocessor computers. Outlined its tasks, basic principles and features on computers with different communication media.

Keywords: testing, multiprocessor computer, distributed system.

Литература

1. Е.П.Велихов, В.Б.Бетелин, С.Г.Бобков, В.А.Галатенко, А.Н.Годунов, А.И.Грюнталь, А.Г.Кушниренко, П.Н.Осипенко, С.Г.Романюк, С.А.Сидоров. Аппаратно-программная платформа «Багет». Концепция и возможности. Издание первое. М., НИИСИ РАН, 2004, 384 стр.
2. ГОСТ Р МЭК 821-2000.
3. Н.Слепов. RapidIO – коммуникационная структура последовательного типа. «Электроника: Наука, Технология, Бизнес». 2006 г., №5, 76-89
4. Е.В.Книга, И.О.Жаринов. Алгоритм тестирования мультипроцессорных многомодульных бортовых цифровых вычислительных систем интегрированной модульной авионики. «Вестник науки Сибири». 2014. № 2 (12), 107-114.
5. П.А.Чибисов. Тестирование микропроцессоров и их rtl-моделей приложениями пользователя под ОС linux. «Программные продукты и системы», 2012, № 3, 112-116.

О соотношении между классами солнц в несимметрично нормированных пространствах

А.Р. Алимов

ФГБОУ ВО « Московский государственный университет имени М.В.Ломоносова», Механико-математический факультет, Москва, Россия, E-mail : alexey.alimov@gmail.com

Аннотация: Известный результат Л.П. Власова о соотношении между классами δ- и γ-солнц обобщается на случай пространств с несимметричной нормой.

Ключевые слова: солнце, несимметрично нормированное пространство, наилучшее приближение

Функция $d: X \times X \to \mathsf{R}_+$ называется *несимметричной метрикой*, если

(1) для любых $x, y \in X$ $d(x,y) = 0$ если и только если $x = y$;

(2) $d(x,z) \le d(x,y) + d(y,z)$ для любых $x, y, z \in X$.

Пара (X, d) называется *несимметричным метрическим пространством* (asymmetric metric space).

Несимметричная норма $\|\cdot\|$ на линейном вещественном пространстве X определяется как $\| x - y \| = d(y,x)$ (в записи $\| x - y \|$ важен порядок членов — в общем случае $\| x - y \| \ne \| y - x \|$). Неравенство треугольника для несимметричной нормы записывается стандартным образом: $\| x - y \| \le \| x - z \| + \| z - y \|$. Термин несимметричная норма был предложен М. Г. Крейном в 1938 г. при исследовании экстремальных вопросов, связанных с проблемой моментов Маркова. Несимметричные расстояния возникают в задачах выпуклого анализа, задачах одностороннего приближения, геометрии банаховых пространств, теории универсальных пространств, а также в прикладных задачах теоретической информатики при анализе сложности программ (см. [4], [5]). По поводу результатов общей теории несимметрично нормированных пространств, а также вопросов функционального анализа и теории приближений в пространствах с несимметричной нормой и метрикой мы отсылаем читателя к монографии С. Кобзаша [2].

На линейном пространстве с несимметричной нормой $(X, \|\cdot\|)$ рассматриваются ``замкнутые'' шары

$$B^+(x,r) = \{y \mid \| y - x \| \le r\},$$

$$B^-(x,r) = \{y \mid \| x - y \| \le r\}.$$

Аналогично определяются ``открытые'' шары $Bo^+(x,r)$, $Bo^-(x,r)$, а также сферы $S^+(x,r)$ и $S^-(x,r)$. Для ``+''-объектов индекс ``+'' мы будем иногда опускать: $B(x,r) = B^+(x,r)$.

Восходящая топология (f-топология, forward topology) τ^+ на пространстве X, индуцируемая метрикой d, порождается f-шарами $Bo^+(x,\varepsilon) := \{y \mid d(x,y) < \varepsilon\}$, $x \in X$, $\varepsilon > 0$. Соответственно, *нисходящая топология* (b-топология, backward topology) τ^- определяется b-шарами $Bo^-(x,\varepsilon)$. В конечномерном случае топологии τ^+ и τ^- эквивалентны, в бесконечномерном случае они могу быть различны.

Отметим следующий результат. Пусть X — линейное пространство с несимметричной нормой. Тогда имеют место следующие соотношения:

$$B(x,r) \subset B(x',r') \Leftrightarrow \| x - x' \| \le r' - r;$$

$$B(x,r) \subset B(x',r') \Leftrightarrow \| x - x' \| < r' - r.$$

Аналогичные утверждения имеют место для b-шаров:

$$B^-(x,r) \subset B^-(x',r') \Leftrightarrow \| x' - x \| \le r' - r;$$

$$B^-(x,r) \subset B^-(x',r') \Leftrightarrow \| x' - x \| < r' - r.$$

В самом деле, если $\| x' - x \| \le r' - r$, то для любого элемента $y \in B(x,r)$ имеем $\| y - x' \| \le \| y - x \| + \| x - x' \| \le r + r'$, т.е. $y \in B(x',r')$. Обратно, пусть $B(x,r) \subset B(x',r')$. Через точки x, x' проведем прямую (если $x = x'$, то можно взять любую из таких прямых) до пересечения со сферой $S(x,r)$ в точке y, для которой выполнено $x \in [y, x']$. Тогда $\| y - x' \| = \| y - x \| + \| x - x' \|$, откуда

$\| x-x' \| = \| y-x' \| - \| y-x \| = \| y-x' \| - r$, а так как $y \in B(x,r) \subset B(x',r')$, то $\| y-x' \| \le r'$, откуда $\| x-x' \| \prec r'-r$. Доказательство параллельного утверждения полностью аналогично.

Величиной наилучшего приближения или *расстоянием* от заданного элемента x линейного несимметрично нормированного пространства X до заданного непустого множества $M \subset X$ называется величина $\rho(x,M) := \inf_{y \in M} \| y-x \|$.

Отметим следующие соотношения (см., например, [2, предложение 1.1.12]):

$$\rho(z,M) \le \| x-z \| + \rho(x,M),$$
$$-\| z-x \| \le \rho(z,M) - \rho(x,M) \le \| x-z \|. \quad (1)$$

Последовательность (x_n) из пространства с несимметричной метрикой (X,d) *сходится вперед* (*f-сходится*) к $x \in X$, если $d(x,x_n) \to 0$ при $n \to \infty$. Последовательность (x_n) *сходится назад* (*b-сходится*) к x, если $d(x_n,x) \to 0$ при $n \to \infty$.

Последовательность (x_n) в пространстве с несимметричной метрикой (X,d) называется *f-последовательностью Коши* (*f-ПК, forward Cauchy sequence*), если для любого $\varepsilon > 0$ найдется $n_0 \in \mathbb{N}$ такое, что при любых $n_0 \le n \le m$ выполнено $\rho(x_n,x_m) < \varepsilon$. Пространство X *f-полно*, если любая f-ПК из X является f-сходящейся (к элементу из X). Отметим, что пространство X *f-полно* если и только если любая f-ПК из X содержит сходящуюся подпоследовательность ([3, лемма 4.3]). Аналогично определяется b-полнота и b-последовательность Коши (b-ПК).

Дж. Коллинс и Й. Циммер [3, пример 3.6]), а также Дж. Кэлли (см. [2]) построили пример пространства с несимметричной метрикой и f-сходящейся последовательности в нем, не являющейся f-ПК.

Множество $M \subset X$ называется δ-*солнцем*, если для любой точки $x \notin M$ найдется последовательность (z_n), $z_n \ne x$, такая, что

$$\| x-z_n \| \to 0, \lim_{n \to \infty} \frac{\rho(z_n,M) - \rho(x,M)}{\| x-z_n \|} \to 1. \quad (2)$$

Множество $M \subset X$ называется γ-*солнцем*, если для любых $x \notin M$ и $r > 0$ существует последовательность $(z_n) \subset X$, такая, что

$$\rho(z_n,M) - \rho(x,M) \to r, \qquad \| x-z_n \| = r. \quad (3)$$

``Солнечные" свойства множеств важны в задачах характеризации элемента наилучшего приближения, устойчивости операторов наилучшего и почти наилучшего приближения, имеют приложения в задачах геометрической оптики и в задачах, связанными с уравнениями Гамильтона—Якоби. Понятия δ- и γ-солнц введены Л. П. Власовым (см. [1]).

Основной результат работы устанавливает связь между классами δ-и иммγ-солнц в пространствах с несимметричной нормой.

Теорема. *Пусть X — линейное несимметрично нормированное пространство. Тогда в X любое γ-солнце является δ-солнцем.*

Если X b-полно, то любое δ-солнце является γ-солнцем.

В симметричном случае результат теоремы 1 установлен Л. П. Власовым (см., например, [1]). Власов также построил пример неполного линейного нормированного пространства, содержащего δ-солнце, не являющееся γ-солнцем.

Замечание. Приведенные выше определения δ- и γ-солнц соответствуют b-сходимости. Для f-сходимости определения аналогичны, при этом условие (2) в определении γ-солнца меняется на

$$\rho(x,M) - \rho(z_n,M) \to r, \qquad \| z_n - x \| = r, \quad (4)$$

а условие (3) заменяется на условие

$$\| z_n - x \| \to 0, \quad \lim_{n \to \infty} \frac{\rho(x,M) - \rho(z_n,M)}{\| z_n - x \|} \to 1. \quad (5)$$

Доказательство. Пусть $x \notin M$, $M \subset X$ — γ-солнце. По определению γ-солнца для любого $n \in \mathbb{N}$ найдется элемент $z_n \in X$ такой, что $\| x-z_n \| = \frac{1}{n}$ и

$$\rho(z_n,M) - \rho(x,M) \quad \frac{1}{n} - \frac{1}{n^2}.$$

Отсюда, так как $\| x-z_n \| = \frac{1}{n}$, то

$$\frac{\rho(z_n,M) - \rho(x,M)}{\| x-z_n \|} \quad 1 - \frac{1}{n}.$$

Далее, по неравенсту (1) имеем $\rho(z_n,M) \le \| x-z_n \| + \rho(x,M)$, что окончательно дает

$$\frac{\rho(z_n,M) - \rho(x,M)}{\| x-z_n \|} \to 1, \qquad n \to \infty,$$

т.е. M — δ-солнце.

Обратно, пусть M является δ-солнцем. Возьмем произвольные $x \notin M$, $r > 0$, $\sigma > 1$. Следуя Р. Фелпсу и Л. П. Власову, ведем порядок в шаре $V := B^-(x,r)$ следующим образом: положим $z \prec z'$, если

$$\| z - z' \| \le \sigma(\rho(z', M) - \rho(z, M)).$$

Антисимметричность и транзитивность отношения порядка очевидны. Покажем, что $\prec$ удовлетворяет условиям леммы Цорна. Пусть (z_α) — цепь в V. При $\alpha \le \alpha'$ имеем

$$\| z_\alpha - z_{\alpha'} \| \le \sigma(\rho(z_{\alpha'}, M) - \rho(z_\alpha, M))$$

По условию (6) числовая цепь $\rho(z_\alpha, M)$ монотонна по α, а из неравенства $\rho(z, M) \le \| x - z \| + \rho(x, M)$ и включения $z \in B^-(x,r)$ следует, что она ограничена. Соответственно, $\rho(z_\alpha, M)$ сходится. Это влечет, что цепь (z_α) является b-ПК. Поскольку X b-полно, то цепь (z_α) b-сходится к некоторому $z \in V$, т.е. $\| v - z_\alpha \| \to 0$.

Ясно, что

$$\| z - z_\alpha \| \le \sigma(\rho(z_\alpha, M) - \rho(z, M)),$$

так что $z_\alpha \prec z$ для всякого α, т.е. цепь (z_α) обладает верхней гранью z. Таким образом порядок $\prec$ удовлетворяет условиям леммы Цорна, согласно которой существует максимальный элемент $v \in V : v \ne x$.

Предположим, что $v \in B^-(x,r)$. Поскольку $\rho(v, M)$ $\rho(x, M) > 0$, то $v \notin M$. По определению δ-солнца найдется последовательность $z_n \ne v$, $\| v - z_n \| \to 0$, для которой

$$(6)\quad \frac{\rho(z_n, M) - \rho(v, M)}{\| v - z_n \|} \to 1, \qquad n \to \infty.$$

Следовательно, поскольку $\sigma > 1$ и $v \in B^-(x,r)$, то при некотором n_0 для всех n n_0 имеем $z_n \in V$,

$$\| v - z_n \| \le \sigma(\rho(z_n, M) - \rho(v, M)),$$

что противоречит максимальности v.

Итак, v лежит на границе V, т.е. $v \in S^-(x,r)$. Условие $x \prec v$ означает по определению, что

$$\| x - v \| \le \sigma(\rho(v, M) - \rho(x, M)).$$

Положим $v_\sigma = v$. В силу (6) имеем

$$\frac{1}{\sigma} \le \frac{\rho(v_\sigma, M) - \rho(x, M)}{\| x - v_\sigma \|} \le 1,$$

что дает

$$\frac{\rho(v_\sigma, M) - \rho(x, M)}{\| x - v_\sigma \|} \to 1, \qquad \sigma \to 1 + 0.$$

Поскольку $\| v_\sigma - x \| = r$, то M по определению является γ-солнцем, что и требуется.

Работа выполнена при поддержке РФФИ (грант 16-01-00295).

On a relation between classes of suns in asymetrically normed spaces

A.R. Alimov

Abstract: The well-known Vlasov's result on the relation between classes of δ- and γ-suns is extended to the case of asymmetrically normed linear spaces.

Keywords: sun, linear space with asymmetric norm, best approximation.

Литература

1. А. Р. Алимов, И. Г. Царьков. Связность и солнечность в задачах наилучшего и почти наилучшего приближения. «УМН», 2016, том. 1 (427), с. 3-84.

2. S. Cobzaş. Functional analysis in asymmetric normed spaces. Birkhäuser, Basel, 2013.

3. J. Collins, J. Zimmer. An asymmetric Arzelà—Ascoli theorem. «Topology and its Applications», vol. 154, 2007, pp. 2312-2322.

4. R. C. Flagg, R. D. Kopperman. The asymmetric topology of Computer Science. in Mathematical Foundations of Programming Semanticsю Springer, 1993, pp. 544-553.

5. G. Mayor, O. Valero, Aggregation of asymmetric distances in Computer Science. «Information Sciences», vol.180. No. 6, 2010, pp. 803-812.