

Математическая модель многотиповых марширующих квадратов и алгоритм их реализации

Н. С. Васильев, Н. Н. Иванова

Аннотация—Классический алгоритм марширующих квадратов (Marching Squares) ограничен бинарными состояниями вершин ячейки и не позволяет напрямую обрабатывать гетерогенные воксельные данные с несколькими типами материалов. В работе предлагается расширение этого подхода – модель многотиповых марширующих квадратов (MTMS), которая увеличивает пространство топологических конфигураций ячейки с классических 16 до 512 состояний за счет поддержки четырех типов материалов. Для приведения глобальных идентификаторов материалов к локальному пространству таблицы конфигураций разработан метод нормализации, основанный на модульной арифметике. Чтобы устранить диагональную неоднозначность, вводится бинарный флаг, вычисляемый через сравнение весовых сумм плотностей на противоположных диагоналях ячейки. Доказано, что предложенное кодирование через индекс кейса обеспечивает однозначное соответствие между состояниями ячейки и элементами таблицы конфигураций. Разработанная модель позволяет извлекать полигональные границы за один проход по сетке без глобальной синхронизации, что критически важно для параллельных реализаций на GPU.

Ключевые слова—воксели, компьютерная графика, марширующие квадраты, параллельные алгоритмы.

I. ВВЕДЕНИЕ

В современных научных и инженерных приложениях широко используются регулярные сетки (решетки), в узлах которых определены дискретные скалярные поля: поле плотности (степень заполненности) и поле идентификаторов материалов. Для преобразования такого объемного представления в полигональную сетку, пригодную для рендеринга и дальнейшей обработки, традиционно применяется алгоритм Marching Squares (MS) и его трехмерное обобщение Marching Cubes (MC) [1].

В классическом варианте алгоритма MS каждая из четырех вершин квадратной ячейки может находиться в одном из двух состояний – «пусто» или «заполнено», что формирует пространство из $2^4 = 16$ топологических конфигураций, полностью описывающих границу изолинии для одного типа материала. Данный подход

эффективен для извлечения границ в однородных средах.

Однако в прикладных областях, таких как научная визуализация, медицинское моделирование (например, построение трехмерных поверхностей по данным мультиспектральной магнитно-резонансной томографии или компьютерной томографии) и геоинформационные системы, все чаще встречаются гетерогенные воксельные данные, в которых различные вершины одной ячейки могут соответствовать разным типам материалов [2], [3]. Прямое применение классического алгоритма MS к таким данным требует либо многократных независимых проходов (по одному на каждый тип материала), что приводит к линейному росту вычислительных затрат, либо адаптации алгоритма с расширением множества состояний вершины с бинарного $\{0, 1\}$ до четырехзначного $\{0, 1, 2, 3\}$. Последнее, однако, вызывает экспоненциальный рост числа базовых конфигураций ($4^4 = 256$ для четырех материалов), усугубляет проблему неоднозначности триангуляции в конфигурациях типа «шахматная доска» и требует отказа от предвычисленной таблицы конфигураций, что критично для систем реального времени [4], [5].

При реализации алгоритмов на современных параллельных архитектурах (многоядерных центральных процессорах (CPU) и графических процессорах (GPU)) необходимо обеспечивать независимую обработку ячеек (отсутствие гонок данных), работу с ограниченным объемом разделяемой памяти (порядка десятков килобайт на блок потоков) и высокую степень параллелизма без глобальной синхронизации [4], [5].

В исследованиях последних лет предложены различные подходы к решению проблемы многоматериального извлечения границ. Например, J. Zhang et al. для снижения топологических ошибок использовали знаковые ограничения, но предложенный ими двухэтапный метод основан на последовательной обработке данных, что ограничивает его применимость в системах реального времени [6]. S. Li et al. в ходе геоинформационного моделирования пришли к выводу, что без детерминированного разрешения диагональных неоднозначностей на стыке разнородных сред неизбежно появление разрывов и перекрытий [7]. T. Chakkour обосновал необходимость нормализации глобальных идентификаторов материалов на уровне локальных ячеек для корректной конечно-элементной дискретизации [8]. Расширение таблиц конфигураций, которое было предложено L. Wang et al., действительно

Статья получена 10 мая 2026.

Васильев Никита Сергеевич, Чувашский государственный университет, Россия (e-mail: nvjojarulesytbodyk@gmail.com; ORCID: 0009-0004-9302-2379).

Иванова Надежда Николаевна, Чувашский государственный университет, Россия (e-mail: niva_mail@mail.ru; ORCID: 0000-0001-7130-8588).

позволяет устранить визуальные артефакты, однако сам алгоритм ориентирован только на бинарные поля и не обеспечивает масштабируемости для сценариев с четырьмя материалами без значительного роста вычислительной сложности [9]. Теоретические основы локального разрешения неоднозначностей через дискретные топологические инварианты наиболее полно систематизированы в современных работах по вычислительной геометрии цифровых изображений [10].

Эти подходы демонстрируют, что направление многоматериальной изоповерхностной реконструкции активно развивается, однако вопрос построения компактной таблицы конфигураций для двумерного случая MS с поддержкой четырех материалов и однозначным разрешением диагональной неоднозначности без привлечения глобальной синхронизации остается открытым.

Цель исследования – разработка математической модели и алгоритма многотиповых марширующих квадратов (MTMS) для генерации полигональных сеток на основе гетерогенных воксельных данных.

Для достижения цели требуется, во-первых, расширение пространства топологических конфигураций ячейки с 16 до 512 состояний для поддержки четырех типов материалов; во-вторых, создание метода нормализации глобальных идентификаторов материалов к локальному пространству таблицы конфигураций; в-третьих, формализация правила разрешения диагональной неоднозначности на основе весовых коэффициентов вершин; и, в-четвертых, построение самой таблицы конфигураций с описанием алгоритма ее генерации.

II. ФОРМАЛИЗАЦИЯ ЗАДАЧИ

A. Входные данные и модель ячейки

Входными данными алгоритма являются два дискретных скалярных поля, определенных на двумерной решетке размером $X \times Y$:

- 1) поле плотности $D(x, y)$, где каждое значение $d \in [0, 0; 1, 0]$ характеризует степень заполненности узла сетки материалом (рис. 1);
- 2) поле материалов $M(x, y)$, где каждое значение $m \in \mathbb{Z}^+$ соответствует глобальному идентификатору типа материала (рис. 2).

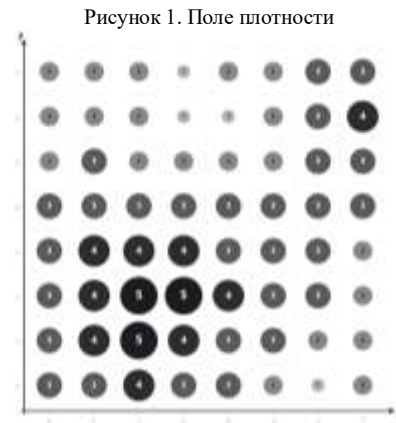
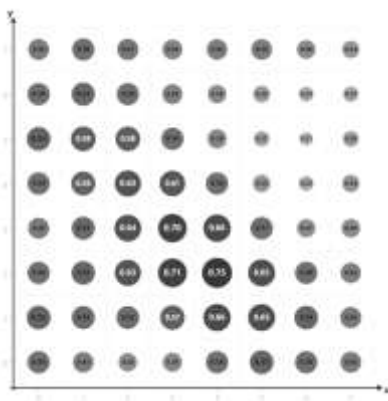


Рисунок 2. Поле материалов

Базовым элементом обработки воксельных данных, заданных на регулярной решетке, является двумерная ячейка $C_{i,j}$, определяемая четырьмя узлами (вершинами)

$$C_{i,j} = \{v_{i,j}, v_{i+1,j}, v_{i+1,j-1}, v_{i,j-1}\},$$

где индексы $i = 0, 1, \dots, X - 1$ и $j = 0, 1, \dots, Y - 1$ задают положение нижнего левого узла ячейки в глобальной решетке.

Для локальной нумерации вершин внутри ячейки используется индекс $k \in \{0, 1, 2, 3\}$. Соответствие между глобальными координатами и локальным индексом k задается порядком обхода вершин (рис. 3):

$$v_0 = v_{i,j}, v_1 = v_{i+1,j}, v_2 = v_{i+1,j-1}, v_3 = v_{i,j-1}.$$

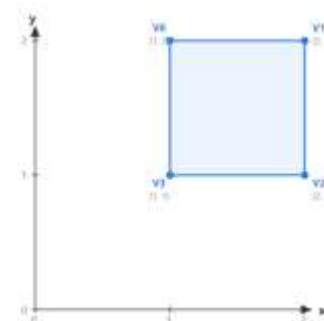


Рисунок 3. Нумерация вершин ячейки

Для каждой вершины ячейки с локальным индексом k определены два атрибута: тип материала m_k и коэффициент плотности d_k :

$$v_k = \{m_k, d_k\}.$$

B. Математическая модель индексации конфигураций

В предлагаемой модели каждой вершине с локальным индексом k ставится в соответствие один из $N_t = 4$ нормализованных типов материалов $t_k \in \{0, 1, 2, 3\}$. Для конфигураций типа «шахматная доска», в которых противоположные вершины имеют одинаковые типы материалов, существуют два возможных способа триангуляции. Такая ситуация называется диагональной неоднозначностью. Для ее устранения вводится дополнительный бинарный флаг $w \in \{0, 1\}$, вычисляемый на основе сравнения сумм плотностей противоположных вершин.

Полное пространство состояний ячейки формируется как комбинация N_t^4 вариантов распределения типов материалов по четырем вершинам и двух вариантов

значения флага.

Индекс топологического кейса K вычисляется как полином в системе счисления с основанием N_t :

$$K = \sum_{k=0}^3 t_k N_t^k + w N_t^4. \quad (1)$$

В выражении (1) первое слагаемое соответствует распределению материалов по вершинам, второе – значению флага неоднозначности.

После подстановки $N_t = 4$ формула (1) принимает вид:

$$K = t_0 + 4t_1 + 16t_2 + 64t_3 + 256w. \quad (2)$$

Индекс K может принимать 512 значений (от 0 до 511). Полученное пространство из 512 конфигураций позволяет обрабатывать границы между несколькими материалами за один проход по всем ячейкам решетки.

С. Структура выходных данных

Выходными данными алгоритма является полигональная сетка V . Для каждой обрабатываемой ячейки генерируется набор треугольников, аппроксимирующих границы материалов.

Максимальное количество вершин сетки (N_{max}) определяется произведением количества вершин в треугольнике, типов материалов (N_t) и максимального количества треугольников на один тип материала ($N_{triangle}$), т.е.

$$N_{max} = 3N_t N_{triangle} = 36.$$

D. Таблица топологических конфигураций

Основным элементом алгоритма является предварительно вычисленная таблица конфигураций (lookup table) T , задающая отображение каждого возможного значения индекса K в набор треугольников для каждого из четырех типов материалов. Таблица генерируется однократно на CPU в момент инициализации системы.

Таблица T представляет собой четырехмерный тензор, линеаризованный в одномерный массив для обеспечения эффективного произвольного доступа. Размерность тензора:

- 1) измерение 1 ($N_K = 512$): индекс кейса K ;
- 2) измерение 2 ($N_t = 4$): тип материала (блок);
- 3) измерение 3 ($N_{triangle} = 3$): индекс треугольника в блоке от 0 до 2;
- 4) измерение 4 ($N_{vertex} = 3$): кортеж из трех индексов опорных точек $[0..11]$, задающих вершины треугольника.

Таким образом, общее количество целочисленных элементов в таблице, определяемое произведением размерностей всех ее измерений, составляет 18 432.

Этот объем памяти (при использовании 4-байтовых целых чисел равен 72 КБ) фиксирован и не зависит от размера входных данных, что позволяет разместить таблицу в быстром кэше L2 или L3 современных CPU.

Линейный доступ к элементу таблицы организован по формуле

$$index = K \cdot S_K + k \cdot S_b + triangle \cdot 3 + vertex,$$

где K – индекс топологического кейса из (1); $k \in \{0, 1, 2, 3\}$ – локальный индекс вершины ячейки; $triangle \in \{0, 1, 2\}$ – номер треугольника в блоке;

$vertex \in \{0, 1, 2\}$ – номер вершины треугольника. Константы шагов имеют следующие значения $S_K = 36$ (шаг между кейсами) и $S_b = 9$ (шаг между блоками для разных вершин); они определены в скрипте генерации таблицы.

Для однозначной интерпретации индексов вершин, записываемых в таблицу конфигураций, вводится локальный базис из 12 опорных точек $P = \{p_0, \dots, p_{11}\}$, жестко привязанных к геометрии квадратной ячейки (рис. 4–6). Точки разделены на три топологические группы:

1) *угловые вершины* ($p_0, p_1, p_2, p_3 = v_0, v_1, v_2, v_3$): соответствуют углам ячейки (0 и 1 – верхние левый и правый, 2 и 3 – нижние правый и левый);

2) *точки на ребрах* (p_4, p_5, p_6, p_7): вычисляются методом линейной интерполяции на пересечении границы материалов с верхним, правым, нижним и левым ребрами соответственно.

3) *диагональные точки* (p_8, p_9, p_{10}, p_{11}): вычисляются путем интерполяции от угловых вершин к геометрическому центру ячейки с учетом весовых коэффициентов плотностей (используются для разрешения седловых неоднозначностей).

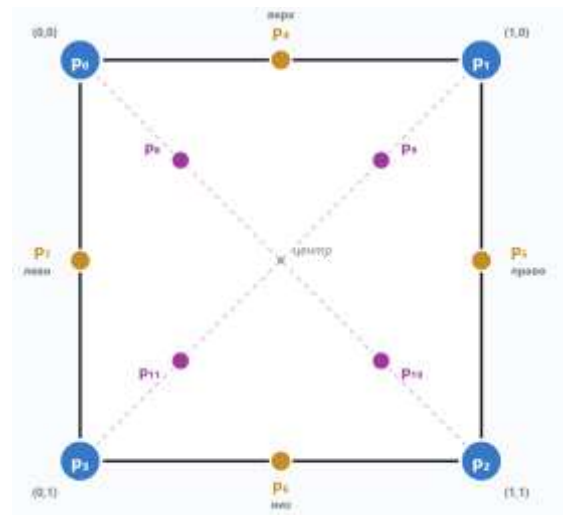


Рисунок 4. Локальный базис из 12 опорных точек ячейки MTMS

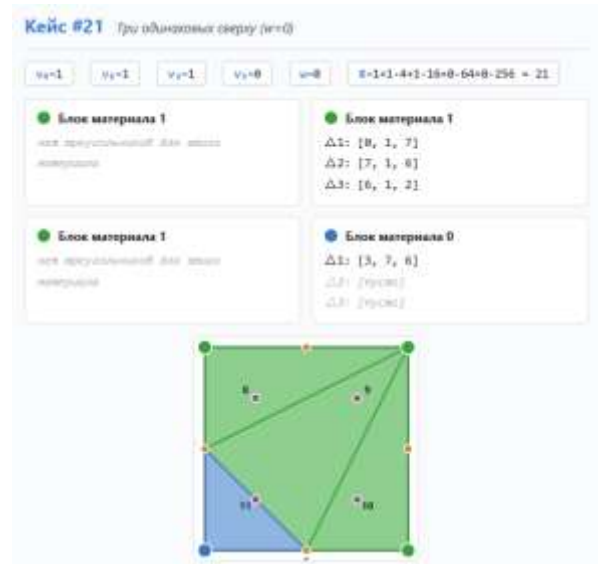


Рисунок 5. Содержимое тензора кейса 21

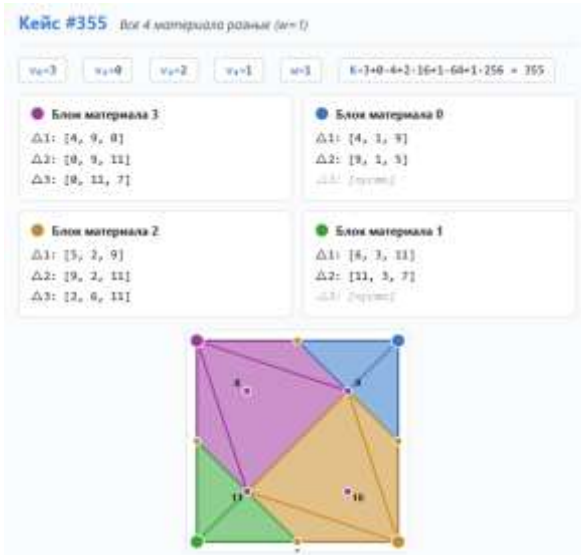


Рисунок 6. Содержимое тензора кейса 355

В процессе обработки ячейки выполняется обращение к фрагменту таблицы конфигураций, соответствующему вычисленному значению индекса K . Для каждого из четырех материальных блоков производится последовательное считывание не более трех треугольников. В случае, когда кортеж индексов опорных точек отличен от нуля (маркер пустого слота), осуществляется выборка соответствующих координат из массива интерполированных значений P и построение итогового полигона. Такой формат хранения гарантирует фиксированный шаг чтения данных и отсутствие условных переходов на этапе сборки сетки, что критически важно для эффективной загрузки в кэш-память и векторизации вычислений при адаптации алгоритма к архитектуре GPU.

III. АЛГОРИТМ НОРМАЛИЗАЦИИ ИДЕНТИФИКАТОРОВ МАТЕРИАЛОВ

Чтобы использовать таблицы конфигураций T , глобальные идентификаторы материалов нужно привести к локальному виду $\{0, 1, 2, 3\}$. Это преобразование выполняет функция M_{norm} , которая отображает набор глобальных идентификаторов ячейки $M_C = \{m_0, m_1, m_2, m_3\}$ ($m_k \in \mathbb{Z}^+$) в набор локальных $\{t_0, t_1, t_2, t_3\}$ ($t_k \in \{0, 1, 2, 3\}$):

$$\{t_0, t_1, t_2, t_3\} = M_{norm}(m_0, m_1, m_2, m_3)$$

Алгоритм, рассчитанный на выполнение на GPU, состоит из двух этапов: сначала вычисляется динамический модуль, затем глобальные идентификаторы материалов последовательно приводятся к диапазону $\{0, 1, 2, 3\}$ с помощью операции взятия остатка от деления.

Псевдокод нормализации идентификаторов материалов ячейки:

```

Функция NORMALIZE_CELL_IDS(blocks[4], T):

// Шаг 1:
// Сдвиг на +1 для исключения нулевых значений
// (необходимо для корректной
// работы модульной арифметики)
FOR I ← 0 TO 3 DO

```

```

    vals[i] ← blocks[i] + 1
END FOR

// Шаг 2: Итеративное сжатие диапазона значений
// Выполняется 4 итерации для гарантии сходимости
FOR iter ← 1 TO 4 DO
    // Нахождение текущего максимума в наборе
    max_val ← MAX(vals[0], vals[1], vals[2], vals[3])

    // Применение модульного отображения
    // с сохранением +1
    FOR I ← 0 TO 3 DO
        vals[i] ← (vals[i] MOD max_val) + 1
    END FOR
END FOR

// Шаг 3: Финальное приведение к диапазону [0; T-1]
max_val ← MAX(vals[0], vals[1], vals[2], vals[3])

FOR I ← 0 TO 3 DO
    norm[i] ← vals[i] MOD max_val
END FOR

RETURN norm

```

Пример работы алгоритма. Пусть $M_C = \{5, 12, 5, 27\}$. Данный набор содержит три уникальных типа. Результат последовательного применения алгоритма нормализации показан в таблице 1.

Таблица 1 – Итерации алгоритма нормализации

Этап	t_0	t_1	t_2	t_3
Входные значения	5	12	5	27
Инициализация	6	13	6	28
Итерация 1	7	14	7	1
Итерация 2	8	1	8	2
Итерация 3	1	2	1	3
Финальные значения	1	2	1	0

Таким образом, после выполнения алгоритма получены нормализованные значения материалов:

$$M_{norm}(5, 12, 5, 27) = \{1, 2, 1, 0\}.$$

IV. РАЗРЕШЕНИЕ ДИАГОНАЛЬНОЙ НЕОДНОЗНАЧНОСТИ

A. Постановка проблемы неоднозначности

Ячейка $C_{i,j}$ называется *неоднозначной*, если ее вершины удовлетворяют условию $t_0 = t_2 \neq t_1 = t_3$, где t_k – нормализованный тип материала в вершине с локальным индексом k . В такой конфигурации границы между материалами могут быть проведены по одной из двух диагоналей квадрата. Варианты отображений для различных w показаны на рис. 7 и 8.



Рисунок 7. Варианты конфигураций MTMS для $w = 0$ Рисунок 8. Варианты конфигураций MTMS для $w = 1$

Если не ввести дополнительное правило, в соседних ячейках возможно использование разных диагоналей, что приводит к несовместимости сеток на общей границе и, как следствие, к визуальным артефактам (щелям или наложениям).

В. Метод разрешения

Для осуществления однозначного выбора диагонали вводится бинарный флаг $w \in \{0, 1\}$, значение которого определяется на основе поля плотности $D(x, y)$. Сначала вычисляются суммы плотностей на противоположных диагоналях:

$$\begin{aligned} d_{0,2} &= d_0 + d_2; \\ d_{1,3} &= d_1 + d_3, \end{aligned}$$

где d_k – коэффициент плотности в вершине с локальным индексом k .

Бинарный флаг вычисляется путем сравнения полученных сумм:

$$w = \begin{cases} 0, & \text{если } d_{0,2} \geq d_{1,3}; \\ 1, & \text{если } d_{0,2} < d_{1,3}. \end{cases}$$

В вырожденном случае, когда $d_{0,2} + d_{1,3} = 0$ (все вершины имеют нулевую плотность), рекомендуется использовать значение $w = 0$ по умолчанию.

Введение флага w удваивает пространство топологических конфигураций для каждого набора типов материалов и отражается в структуре таблицы T . Таблица организуется в два слоя: T_0 для конфигурации с $w = 0$ (рис. 4) и T_1 для $w = 1$ (рис. 5). Выбор слоя выполняется на этапе вычисления индекса K по формуле (2).

V. АЛГОРИТМ ГЕНЕРАЦИИ ТАБЛИЦЫ КОНФИГУРАЦИЙ

Генерация таблицы T выполняется однократно на центральном процессоре при инициализации системы. Алгоритм состоит из трех последовательных этапов.

Этап 1. Перебор пространства конфигураций и вычисление индекса кейса. Выполняется перебор всех возможных комбинаций типов материалов на четырех вершинах ячейки ($4^4 = 256$ конфигураций) и для каждого из двух значений флага $w \in \{0, 1\}$. Общее число итераций равно 512.

Псевдокод итераций по пространству конфигураций:

Вход: $T \leftarrow 4$ // количество типов материалов
Выход: $\text{table}[512][4][3][3]$ // таблица состояний

```

1:  $Z \leftarrow [0, 0, 0]$  // нулевой кортеж
2:  $\text{stride}_w \leftarrow T^4$  // 256 конфигураций на флаг  $w$ 
3:
4: FOR  $w \in \{0, 1\}$  DO
5:   FOR  $v_0, v_1, v_2, v_3 \in [0; T-1]$  DO
6:      $\text{vals} \leftarrow [v_0, v_1, v_2, v_3]$ 
7:     // Вычисление линейного индекса кейса
8:      $K \leftarrow v_0 + v_1 \cdot T + v_2 \cdot T^2 + v_3 \cdot T^3 + w \cdot \text{stride}_w$ 
9:      $b \leftarrow \text{INITIALIZE\_CELL}(Z)$ 
10:     $U \leftarrow \text{CARDINALITY}(\text{SET}(\text{vals}))$ 
11:    // Переход к классификации
12:    // и заполнению структуры  $b$ 
13:     $\text{ASSIGN\_TOPOLOGY}(b, \text{vals}, U, w)$ 
14:     $\text{table}[K] \leftarrow b$ 
15:   END FOR
16: END FOR
17: RETURN  $\text{table}$ 

```

Этап 2. Топологическая классификация. Для каждой комбинации типов материалов вычисляется базовый индекс K по формуле (2). Значение K служит ключом для записи результатов построения геометрии. Функция ASSIGN_TOPOLOGY выполняет идентификацию паттерна распределения материалов по вершинам и формирует в структуре b запись индексов опорных точек (0–11) для каждого материального блока.

Псевдокод функции ASSIGN_TOPOLOGY :

Вход: $b[4][3][3]$ – структура для записи треугольников
 $\text{vals}[4]$ – нормализованные типы материалов вершин
 $U \in \{1, 2, 3, 4\}$ – число уникальных материалов
 $w \in \{0, 1\}$ – флаг диагональной неоднозначности
Выход: b – структура с назначенными индексами вершин (0..11)

```

1: IF  $U = 1$  THEN
2:   // Однородная ячейка
3:    $b[\text{vals}[0]] \leftarrow [[0, 1, 3], [3, 1, 2], Z]$ 
4:
5: ELSE IF  $U = 2$  THEN
6:   IF  $\text{vals}[0] \neq \text{vals}[2]$  AND  $\text{vals}[1] \neq \text{vals}[3]$  THEN
7:     // Шахматная доска: противоположные вершины одинаковы
8:     IF  $\text{vals}[0] = \text{vals}[1]$  THEN
9:       // Смежные материалы
10:       $b[\text{vals}[0]] \leftarrow [[0, 1, 5], [0, 5, 7], Z]$ 
11:       $b[\text{vals}[2]] \leftarrow [[2, 3, 7], [2, 7, 5], Z]$ 
12:     ELSE
13:       // Противоположные материалы
14:       $b[\text{vals}[1]] \leftarrow [[1, 2, 6], [1, 6, 4], Z]$ 
15:       $b[\text{vals}[3]] \leftarrow [[3, 0, 4], [3, 4, 6], Z]$ 
16:     END IF
17:   ELSE
18:     // Два смежных материала
19:     // Определяется стандартным алгоритмом MS
20:      $\text{APPLY\_STANDARD\_MS}(b, \text{vals}, w)$ 
21:   END IF
22:
23: ELSE IF  $U = 3$  THEN
24:   // Три вершины с одинаковым материалом
25:   // Определяем, какая вершина отличается
26:   IF  $\text{vals}[0] = \text{vals}[1]$  AND  $\text{vals}[1] = \text{vals}[2]$  THEN
27:      $\text{triple} \leftarrow 0$  // отличаются:  $v_3$ 
28:   ELSE IF  $\text{vals}[3] = \text{vals}[1]$  AND  $\text{vals}[1] = \text{vals}[2]$  THEN
29:      $\text{triple} \leftarrow 1$  // отличаются:  $v_0$ 
30:   ELSE IF  $\text{vals}[3] = \text{vals}[0]$  AND  $\text{vals}[0] = \text{vals}[2]$  THEN
31:      $\text{triple} \leftarrow 2$  // отличаются:  $v_1$ 
32:   ELSE IF  $\text{vals}[3] = \text{vals}[0]$  AND  $\text{vals}[0] = \text{vals}[1]$  THEN
33:      $\text{triple} \leftarrow 3$  // отличаются:  $v_2$ 
34:   END IF
35:    $\text{ASSIGN\_TRIPLE\_CASE}(b, \text{vals}, \text{triple}, w)$ 
36:
37: ELSE
38:   //  $U = 4$ : все вершины различны
39:    $\text{ASSIGN\_FULL\_CASE}(b, \text{vals}, w)$ 
40: END IF
41: RETURN  $b$ 

```

Этап 3. Построение кейса. Для каждого кейса K

строятся треугольники для всех $N_t = 4$ типов материалов. Для каждого ребра ячейки проверяется наличие пересечения границы между материалами. При обнаружении пересечения координаты точки пересечения вычисляются методом линейной интерполяции. Для каждой конфигурации определяется кардинальность множества уникальных материалов $U = |\{v_0, v_1, v_2, v_3\}|$. На основе значения U и паттерна равенств соседних или противоположных вершин выбирается один из заранее определенных шаблонов разбиения. Шаблон определяет набор треугольников (до трех на тип материала) и ссылки на опорные точки локального базиса (индексы 0–11). Разрешение седловых неоднозначностей обеспечивается флагом w .

Псевдокод топологической классификации и заполнения ячеек, оптимизированной для дальнейшей работы на GPU:

```
Вход: b[4][3][3], vals[4],  $U \in \mathbb{N}$ ,  $w \in \{0,1\}$ 
Выход: b – структура с назначенными
        индексами вершин (0..11)

1: IF U = 1 THEN
2:   APPLY(b, TEMPLATE_HOMOGENEOUS)
3: ELSE IF U = 2 AND IS_CROSS_PATTERN(vals) THEN
4:   IF vals[0] = vals[1] THEN APPLY(b,
5:   TEMPLATE_ADJACENT)
6:   ELSE APPLY(b, TEMPLATE_OPPOSITE)
7:   END IF
8: ELSE IF U = 3 THEN
9:   p ← IDENTIFY_TRIPLE_PATTERN(vals)
10:  APPLY(b, p, w)
11: ELSE
12:   // U = 4: все вершины различны
13:   APPLY(b, TEMPLATE_FULL_SEGMENTATION, w)
14: END IF
15: RETURN b
```

Важно отметить, что в отличие от классических реализаций, предложенный алгоритм не вычисляет геометрические координаты на этапе построения таблицы. Таблица содержит исключительно топологические связи (индексы опорных точек). Линейная интерполяция координат точек по полям плотностей $D(x, z)$ и материалов $M(x, z)$ выполняется непосредственно во время генерации, что существенно снижает нагрузку на память и обеспечивает линейное масштабирование производительности $O(N)$ относительно объема воксельных данных.

VI. РЕЗУЛЬТАТЫ РАБОТЫ МОДЕЛИ

Прототип алгоритма MTMS и программный визуализатор разработаны на языке программирования Python 3. Вычислительное ядро (генерация таблицы, нормализация, интерполяция) реализовано с использованием библиотеки NumPy, что позволяет выполнить векторизацию операций. Графический интерфейс и рендеринг полигональной сетки выполнены с применением библиотеки Pygame. Исходный код реализации, включая скрипт генерации таблицы из 512 конфигураций и набор тестовых данных, доступен в публичном репозитории [<https://github.com/nvmediagithub/multi-type-marching-squares-python-cpu>].

Ключевой задачей исследования являлась генерация непрерывной полигональной поверхности для сцен, содержащих не менее четырех типов материалов, без

привлечения методов постобработки. В качестве входных данных использовались скалярное поле плотностей $D(x, z)$ (рис. 9) и дискретное поле идентификаторов материалов $M(x, z)$ (рис. 10), состоящее из пяти типов: «вода», «песок», «трава», «скалы», «снег». Результат работы предложенного алгоритма многотиповых марширующих квадратов (MTMS) представлен на рис. 11.



Рисунок 9. Поле плотностей $D(x, y)$ на примере работы MTMS



Рисунок 10. Поле материалов $M(x, y)$ на примере работы MTMS

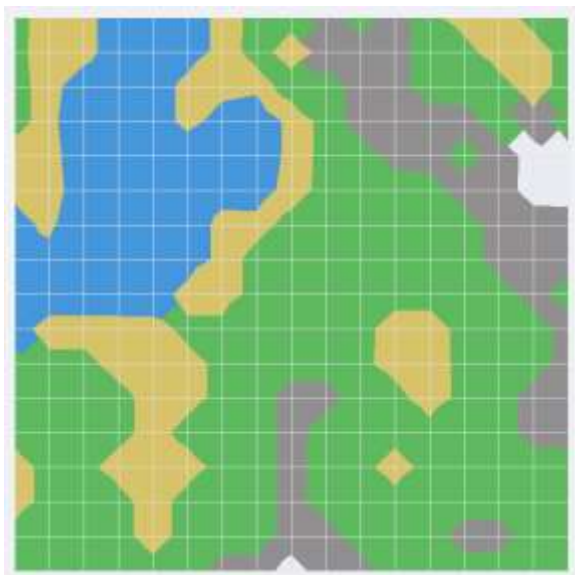


Рисунок 11. Результат работы MTMS

Визуальный анализ подтверждает достижение поставленных целей. Границы раздела материалов формируются непрерывно, без разрывов, наложений

или невалидных топологических соединений. Все четыре типа блоков корректно изолированы за один проход по вычислительной сетке, что подтверждает эффективность предрасчитанной таблицы состояний. Внедрение бинарного флага диагонали w полностью устранило артефакты, характерные для седловых конфигураций в классических реализациях маршрутирующих квадратов.

Полученные результаты свидетельствуют о том, что разработанный метод обеспечивает топологически корректную генерацию многокомпонентных поверхностей и обладает потенциалом для масштабирования на трехмерные воксельные массивы.

Для оценки места предложенного метода MTMS в контексте существующих решений проведен сравнительный анализ ключевых алгоритмических и топологических характеристик современных методов многоматериальной изоповерхностной реконструкции (таблица 2).

Таблица 2 – Сравнение методов многоматериальной изоповерхностной реконструкции

Характеристика	Знаковый MC [6]	Многотипный MC [7]	КЭ-дискретизация [8]	Улучшенный MC [9]	Предлагаемый MTMS
Базовый алгоритм и размерность	3D, Marching Cubes	3D, Multiple Material MC	3D, Прямая сборка КЭ-сетки	3D, Marching Cubes	2D, Marching Squares (с обобщением до 3D)
Поддерживаемое число материалов	2 (бинарное поле)	≥ 3 (фиксировано в модели)	произвольное (глобальные метки)	2 (бинарное поле)	не ограничено глобально; до 4 уникальных локально на ячейку
Пространство конфигураций ячейки (число состояний)	256 (стандартный MC)	не фиксировано	не применяется (прямая сборка)	24 расширенных классов вместо 256 кейсов	512 ($4^4 \times 2$ с учетом флага w)
Алгоритмическая сложность	глобальная: $O(N)$; локальная: $O(1)$; требуется 2 последовательных проходов по сетке	Глобальная: $O(N)$; Локальная: $> O(1)$; требуется обход графа (BFS/DFS) и глобальная синхронизация	Глобальная: $O(N)$; Локальная: $O(1)$; требуется глобальная предобработка и нормализация меток до сборки	глобальная: $O(N)$ (для одного материала); для M материалов деградирует до $O(N \cdot M)$ (множественные проходы)	глобальная: строго $O(N)$ (не зависит от M); локальная: строго $O(1)$; требуется 1 проход, $Span\ O(1)$; идеален для GPU
Разрешение диагональной неоднозначности	не рассматривается (требует дополнительных методов)	эвристическое (через глобальные приоритеты)	не применимо (в рамках КЭ-формализма)	частичное (устранение пустот, но не седловых точек)	детерминированное (бинарный флаг по суммам плотностей)
Пригодность для параллельных вычислений (GPU)	низкая (двухэтапный метод, зависимости)	низкая (глобальный поиск, синхронизация)	ограниченная (сборка КЭ требует синхронизации)	высокая (но только для бинарных полей)	высокая (один проход, отсутствие гонок данных)

Анализ данных таблицы 2 позволяет заключить, что существующие подходы [6–9] не обеспечивают одновременно поддержку произвольного числа материалов, компактность таблицы конфигураций и детерминированное разрешение неоднозначностей, а также требуют глобальной синхронизации или многократных проходов по сетке. В отличие от них, предлагаемый метод MTMS характеризуется строго линейной сложностью $O(N)$, фиксированным пространством конфигураций (512 состояний) и отсутствием межпоточных зависимостей, что делает его пригодным для эффективной реализации на GPU. Таким образом, предложенная модель устраняет указанные ограничения, обеспечивая извлечение полигональных границ для произвольного числа материалов за один

проход по сетке без глобальной синхронизации.

VII. ЗАКЛЮЧЕНИЕ

В данной работе представлена математическая модель MTMS для генерации полигональных сеток на основе гетерогенных воксельных данных.

Основные результаты работы:

1. Разработана математическая модель расширения пространства топологических состояний с 16 до 512 конфигураций при поддержке $N_t = 4$ типов материалов.

2. Предложен метод нормализации идентификаторов материалов $\{m_0, m_1, m_2, m_3\} \rightarrow \{t_0, t_1, t_2, t_3\}$ для приведения глобальных идентификаторов m к локальному пространству состояний таблицы конфигураций.

3. Формализован метод разрешения диагональной неоднозначности через вычисление бинарного флага w на основе диагональных сумм плотностей $d_{0,2}$ и $d_{1,3}$.

4. Разработан алгоритм генерации таблицы конфигураций на CPU. Таблица содержит 18 432 целочисленных элемента.

Практическая значимость исследования определяется возможностью применения разработанной математической модели в следующих областях:

1) научной визуализации: медицинское моделирование, геоинформационные системы, инженерный анализ объемных данных с множественными типами материалов;

2) процедурной генерации: создание виртуальных миров с множественными типами материалов без необходимости множественных проходов по данным;

3) системах реального времени: игровые движки, симуляторы, VR/AR-приложения, где требуется обработка больших объемов гетерогенных данных с частотой обновления более 60 Гц.

Определены следующие направления для развития разработанной математической модели:

1) обобщение до трехмерного случая: расширение модели до Multi-Type Marching Cubes с пространством состояний 2^{12} конфигураций для кубической ячейки;

2) оптимизация алгоритма для работы на GPU: исследование методов размещения часто используемых кейсов в разделяемой памяти (shared memory) и SIMD вычислений внутри вычислительной группы GPU.

БИБЛИОГРАФИЯ

- [1] Lorensen W. E., Cline H. E. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 1987, vol. 21(4), pp. 163–169. DOI: 10.1145/37402.37422.
- [2] Lewiner T., Lopes H., Vieira A.W., Tavares G. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 2003, vol. 8(2), pp. 1–15. DOI: 10.1080/10867651.2003.10487582.
- [3] Chernyaev E.V. Marching cubes 33: Construction of topologically correct isosurfaces. In: CERN Technical Report CN/95-17. Geneva, 1995, 35 p. URL: https://www.researchgate.net/profile/Evgueni-Tcherniaev/publication/2651412_Marching_Cubes_33_Construction_of_Topologically_Correct_Iso-surfaces/links/09e41508ae8909f864000000/Marching-Cubes-33-Construction-of-Topologically-Correct-Iso-surfaces.pdf?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19.
- [4] Smistad E., Falch T.L., Bozorgi M. et al. Medical image segmentation on GPUs – A comprehensive review. *Medical Image Analysis*, 2015, vol. 20(1), pp. 1–18. DOI: 10.1016/j.media.2014.10.012.
- [5] Dyken C., Ziegler G., Theobalt C., Seidel H. P. High-speed marching cubes using histogram pyramids. *Computer Graphics Forum*, 2008, vol. 27(8), pp. 2028–2039. DOI: 10.1111/j.1467-8659.2008.01179.x.
- [6] Zhang J., Li H., Wang L. A two-step surface reconstruction method using signed marching cubes. *Applied Sciences*, 2022, vol. 12(4), 1792. DOI: 10.3390/app12041792.
- [7] Li B., Zhong D., Wang L., Repair of geological models based on multiple material marching cubes. *Mathematics*, 2021, vol. 9, no. 18, p. 2207. DOI: 10.3390/math9182207.
- [8] Chakkour T. Finite element modelling of complex 3D image data with quantification and analysis. *Oxford Open Materials Science*, 2024, vol. 4(1), itae003. DOI: 10.1093/oxfmat/itae003.
- [9] Wang M., Luo H., Cui Q. Three-dimensional reconstruction based on improved marching cubes algorithm. *Journal of Mechanics in Medicine and Biology*, 2020, vol. 20(09), 2040002. DOI: 10.1142/S0219519420400023.
- [10] Peters J.F. Computational Geometry, Topology and Physics of Digital Images with Applications. Cham, Springer, 2020, 356 p. DOI: 10.1007/978-3-030-22192-8.

Mathematical Model of Multi-Type Marching Squares and Algorithm for Their Implementation

N. S. Vasiliev, N. N. Ivanova

Abstract—The classical Marching Squares algorithm is limited to binary vertex states of a cell and does not allow direct processing of heterogeneous voxel data with multiple material types. This paper proposes an extension of this approach – the Multi-Type Marching Squares (MTMS) model, which expands the topological configuration space of a cell from the classical 16 to 512 states by supporting four material types. To map global material identifiers to the local space of the configuration table, a normalization method based on modular arithmetic is developed. To resolve diagonal ambiguity, a binary flag is introduced, computed by comparing weighted density sums on opposite cell diagonals. It is proved that the proposed encoding via case index ensures a one-to-one correspondence between cell states and configuration table entries. The developed model enables extraction of polygonal boundaries in a single pass over the grid without global synchronization, which is critical for parallel GPU implementations.

Keywords—voxels, computer graphics, marching squares, parallel algorithms.

REFERENCES

1. Lorensen W. E., Cline H. E. Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Lorensen, W.E. and Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," ACM SIGGRAPH Computer Graphics, vol. 21, no. 4, pp. 163–169, 1987. DOI: 10.1145/37402.37422.
2. Lewiner, T., Lopes, H., Vieira, A.W., and Tavares, G., "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees," Journal of Graphics Tools, vol. 8, no. 2, pp. 1–15, 2003. DOI: 10.1080/10867651.2003.10487582.
3. Chernyaev, E.V., "Marching Cubes 33: Construction of Topologically Correct Isosurfaces," CERN Technical Report CN/95-17, Geneva, 1995. URL: https://www.researchgate.net/profile/Evgueni-Tcherniaev/publication/2651412_Marching_Cubes_33_Construction_of_Topologically_Correct_Isosurfaces/links/09e41508ae8909f864000000/Marching-Cubes-33-Construction-of-Topologically-Correct-Isosurfaces.pdf
4. Smistad, E., Falch, T.L., Bozorgi, M., et al., "Medical Image Segmentation on GPUs – A Comprehensive Review," Medical Image Analysis, vol. 20, no. 1, pp. 1–18, 2015. DOI: 10.1016/j.media.2014.10.012.
5. Dyken, C., Ziegler, G., Theobalt, C., and Seidel, H.P., "High-Speed Marching Cubes Using Histogram Pyramids," Computer Graphics Forum, vol. 27, no. 8, pp. 2028–2039, 2008. DOI: 10.1111/j.1467-8659.2008.01179.x.
6. Zhang, J., Li, H., and Wang, L., "A Two-Step Surface Reconstruction Method Using Signed Marching Cubes," Applied Sciences, vol. 12, no. 4, article 1792, 2022. DOI: 10.3390/app12041792.
7. Li B., Zhong D., Wang L., "Repair of Geological Models Based on Multiple Material Marching Cubes," Mathematics, vol. 9, no. 18, article 2207, 2021. DOI: 10.3390/math9182207.
8. Chakkour, T., "Finite Element Modelling of Complex 3D Image Data with Quantification and Analysis," Oxford Open Materials Science, vol. 4, no. 1, article itae003, 2024. DOI: 10.1093/oxfmat/itae003.
9. Wang M., Luo H., Cui Q., "Three-Dimensional Reconstruction Based on Improved Marching Cubes Algorithm," Journal of Mechanics in Medicine and Biology, vol. 20, no. 9, article 2040002, 2020. DOI: 10.1142/S0219519420400023.
10. Peters, J.F., Computational Geometry, Topology and Physics of Digital Images with Applications, Cham: Springer, 2020. DOI: 10.1007/978-3-030-22192-8.