Полученные электронные конфигурации всех рассмотренных кристаллов оказались близкими к общепринятой трактовке характерной ионной химической связи, а значения коэффициента компактности совпадают с указанными в литературе. Описанный подход может быть применен для подсчета коэффициентов компактности других двух компонентных кристаллов кубической сингонии.

ЛИТЕРАТУРА

1. *Шаскольская М.П.* Очерки о свойствах кристаллов. – М.: Химия, 1978.
2. *Комарова О.С.* Технология конструкционных материалов. – Минск: Новое знание, 2005.
3. *Шаскольская М.П.* Кристаллография. – М.: Высшая школа, 1976.
4. *Плутенко А.Д., Еремин И.Е., Еремина В.В.* Метод опосредованной визуализации наноструктур // Информатика и системы управления. – 2012. – № 3(33). – С.83-96.
5. *Еремин И.Е., Бартошин А.С.* Опосредованная визуализация наноструктуры кубических оксидных кристаллов // Информатика и системы управления. – 2012. – № 4(34). – С.56-63.
6. SOPRA N&K Database. – http://www.sopra-sa.com.

E-mail:
Еремин Илья Евгеньевич – marinecops@mail.ru;
Бартошин Алексей Степанович – blg.silence.ru@gmail.com

УДК 519.612.2

© 2013 г. **А.Н. Камаев**
(Вычислительный центр ДВО РАН, Хабаровск)

ИССЛЕДОВАНИЕ АЛГОРИТМОВ УПОРЯДОЧИВАНИЯ КОЭФФИЦИЕНТОВ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ, ВОЗНИКАЮЩИХ В ЗАДАЧАХ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Описана процедура решения больших систем линейных алгебраических уравнений с разреженными матрицами, возникающих при проектировании систем компьютерного зрения. Исследована эффективность трех алгоритмов упорядочения изображений для увеличения разреженности системы, предложена модификация алгоритма вложенных сечений для наборов изображений с нерегулярными связями.

Ключевые слова: система линейных алгебраических уравнений, метод наименьших квадратов, метод вложенных сечений, метод параллельных сечений, заполнение матрицы, разреженная матрица.

Введение

Многие задачи компьютерного зрения, связанные с относительной ориентацией больших наборов пересекающихся изображений, приводят к необходимости

решения систем из большого числа линейных алгебраических уравнений. К системам линейных алгебраических уравнений (СЛАУ) приводят как линейные схемы метода наименьших квадратов (МНК), так и нелинейные. В частности, широко применяемый в компьютерном зрении алгоритм Левенберга–Марквардта [1, 2] требует решения таких систем на каждой итерации. Встречающиеся в задачах компьютерного зрения СЛАУ часто имеют разреженные симметричные, положительно определенные матрицы. Степень их разреженности определяется количеством связей между изображениями, рассматриваемыми в задаче. Чем меньше связей, тем более разрежена система. В данной статье рассматривается алгоритм решения СЛАУ с симметричными, положительно-определенными матрицами, учитывающий их разреженность. Исследована эффективность применения трех методов упорядочивания изображений (обратный алгоритм Катхилла–Макки [4], метод вложенных сечений и метод параллельных сечений [5]) для минимизации заполнения матриц систем, возникающих в задачах сшивки панорамных карт (спутниковые карты, карты морского дна, задача навигации на местности и др.).

Структура типичных задач

Рассмотрим две задачи, часто встречающиеся в компьютерном зрении: линейную и нелинейную задачи МНК. Будем считать, что для них задан набор из m изображений.

Линейная задача МНК. Пусть каждое из m изображений является фрагментом одного плоского изображения – панорамы. Все изображения имеют одинаковый масштаб и поворот. Обозначим координаты центров изображений в системе координат панорамы через $\mathbf{c}_i = (c_{ix}, c_{iy})$, $i = 1, 2, \ldots, m$. Зададим два вектора номеров изображений размерности n: $\mathbf{K} = (k_1, k_2, \ldots, k_n)$ и $\mathbf{J} = (j_1, j_2, \ldots, j_n)$, $k_l, j_l \in \{1, 2, .., m\}$, $k_l < j_l$, $l = 1, 2, \ldots, n$, где каждая пара номеров k_l, j_l уникальна ($(k_l, j_l) \neq (k_p, j_p)$, $l, p = 1, 2, \ldots, n$, $l \neq p$). Для каждой пары изображений с номерами k_l, j_l введем вектор $\mathbf{d}_l = \mathbf{d}_{k_l j_l}$, задающий смещение центра изображения $\mathbf{c}_{k_l}$ относительно центра $\mathbf{c}_{j_l}$:

$$\mathbf{d}_l = \mathbf{c}_{k_l} - \mathbf{c}_{j_l}. \qquad (1)$$

Все n векторов $\mathbf{d}_l$ образуют множество $D = \{\mathbf{d}_l\}$. Множество центров изображений $C = \{\mathbf{c}_i\}$ и множество D образуют граф $G(C, D)$. Будем рассматривать случай, когда граф $G(C, D)$ связан. В данной задаче на основе известного множества D необходимо найти множество C.

Решение. Поместим центр m-го изображения в начало системы координат панорамы. Тогда $\mathbf{c}_m = (0, 0)$, а вектор неизвестных $\mathbf{x} = (c_{1x}, c_{1y}, c_{2x}, c_{2y}, \ldots, c_{m-1,x}, c_{m-1,y})$. Уравнения (1) составляют СЛАУ с блочной матрицей $\mathbf{A} = (\mathbf{a}_{li})$, где $i = 1, 2, \ldots, m - 1$ и вектором свободных членов $\mathbf{b} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n)$. Определим блок $\mathbf{a}_{li}$:

$$\mathbf{a}_{li} = \begin{cases} \mathbf{0}^{2\times 2}, & i \neq k_l, j_l, \\ \operatorname{diag}(1, 1), & i = k_l, \\ \operatorname{diag}(-1, -1), & i = j_l. \end{cases}$$

Здесь $\mathbf{0}^{2\times 2}$ – матрица размерности 2×2, состоящая из нулевых элементов. Решение системы $\mathbf{Ax} = \mathbf{b}$ даст искомые координаты центров изображений $\mathbf{c}_i$. В слу-

чае, когда $n = m - 1$, число уравнений в этой системе совпадает с числом неизвестных и равняется $2(m - 1)$, система однозначно разрешима. В общем случае, когда $n > m - 1$, система $\mathbf{Ax} = \mathbf{b}$ будет переопределенной. Тогда вместо точного решения будем искать вектор $\mathbf{x}$, минимизирующий величину e:

$$e = \sum_{l=1}^{n} (\mathbf{d}_l - \mathbf{c}_{k_l} + \mathbf{c}_{j_l})^2 .$$

Вектор $\mathbf{x}$, минимизирующий e, можно найти из нормального уравнения $\mathbf{Qx} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$, где $\mathbf{Q} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ [2]. Матрица $\mathbf{Q} = (\mathbf{q}_{ip})$ – симметричная, положительно определенная и состоит из блоков $\mathbf{q}_{ip}$, $i, p = 1, 2,..., m - 1$, имеющих размерность 2×2. Общее число блоков в этой матрице равно $(m - 1)^2$. Каждый блок $\mathbf{q}_{ip}$ имеет ненулевые элементы только тогда, когда изображения с номерами i и p непосредственно связаны ($\exists l$: $k_l = i$, $j_l = p$ или $k_l = p$, $j_l = i$). В большинстве задач компьютерного зрения количество связей между изображениями имеет порядок $2m \div 8m$, что приводит к разреженной структуре матрицы $\mathbf{Q}$.

Нелинейная задача МНК. В качестве примера нелинейной задачи МНК рассмотрим задачу трехмерной реконструкции. В пространстве расположены n точек $\mathbf{X}_i$, $i = 1, 2,\ldots, n$. Также имеются m камер, каждая из которых характеризуется некоторым вектором параметров $\mathbf{P}_j$, $j = 1, 2,\ldots, m$. Задана вектор-функция $\mathbf{f}$: $\mathbf{x}_{ij} = \mathbf{f}(\mathbf{P}_j, \mathbf{X}_i)$, где $\mathbf{x}_{ij}$ – проекция i-й точки на плоскость проекции j-й камеры (или j-е изображение). В общем случае функция f – нелинейная. Зная проекции $\mathbf{x}_{ij}$, необходимо найти все $\mathbf{X}_i$ и $\mathbf{P}_j$.

В компьютерном зрении для решения подобных задач используется алгоритм Левенберга–Марквардта [1]. Рассматриваемая задача подразумевает два вида разреженности данных: на основе условий задачи и на основе отсутствующих данных. К первому виду разреженности приводит то, что проекция i-й точки на j-е изображение не зависит от параметров k-й камеры и от p-й точки при условии, что $j \neq k$ и $i \neq p$. Разреженность первого вида описана в [2] и приводит к разделению задачи на задачу для параметров камеры и задачу относительно координат пространственных точек на каждой итерации алгоритма Левенберга–Марквардта. Задача для точек имеет линейную вычислительную сложность, а задача относительно параметров камеры сводится к решению блочной СЛАУ следующего вида: $\mathbf{S}\delta_{\mathbf{P}} = \mathbf{e}$ [2], где $\delta_{\mathbf{P}}$ – изменение вектора параметров $(\mathbf{P}_1, \mathbf{P}_2,..,\mathbf{P}_m)$; $\mathbf{e}$ – вектор правых частей; $\mathbf{S} = (\mathbf{s}_{kj})$ – блочная, симметричная, положительно определенная матрица. $\mathbf{S}$ содержит $m \times m$ блоков $\mathbf{s}_{kj}$, $k, j = 1, 2,\ldots, m$. Размерность блока; $\dim\mathbf{s}_{kj} = \dim\mathbf{P}_k \times \dim \mathbf{P}_j$.

Ко второму виду разреженности – разреженности на основе отсутствующих данных – приводит то, что не все $\mathbf{x}_{ij}$ обычно известны. Блок $\mathbf{s}_{kj}$ содержит ненулевые элементы, если есть хотя бы одна точка, видимая на k-м и j-м изображении ($\exists l \in \{1, 2,\ldots, n\}$, такое, что известны $\mathbf{x}_{lk}$ и $\mathbf{x}_{lj}$). В большинстве задач компьютерного зрения, а особенно в задачах сшивки панорамных карт, каждая точка пространства видна лишь на малом количестве изображений, что приводит к сильной разреженности матрицы $\mathbf{S}$ (аналогично матрице $\mathbf{Q}$ в линейной задаче МНК).

Решение систем линейных уравнений

В предыдущем разделе были описаны типичные для компьютерного зрения задачи, приводящие к решению систем вида $\mathbf{Ax} = \mathbf{b}$, где $\mathbf{A}$ – симметричная, положительно определенная разреженная матрица размерности $n \times n$, $\mathbf{A} = (a_{ij})$, $i, j = 1, 2,\ldots, n$. Так как матрица $\mathbf{A}$ симметрична, то для нее существует разложение $\mathbf{A} = \mathbf{LDL}^{\mathrm{T}}$ [3], где $\mathbf{L}$ – нижняя унитреугольная матрица, а $\mathbf{D}$ – диагональная. Для выполнения этого разложения будем использовать алгоритм из [3], модифицировав его для учета разреженности матрицы $\mathbf{L}$. Обозначим номер первого ненулевого элемента i-й строки матрицы $\mathbf{A}$ через s_i: $a_{ik} = 0$, $k < s_i$, тогда $l_{ik} = 0$ [2]. Для каждой строки j матрицы $\mathbf{A}$ введем множество номеров строк, ниже j-й строки: $T_j = \{j + 1, j + 2,\ldots, n\}$. Учитывая номера s_i и множества T_j, перепишем алгоритм из [3] следующим образом:

Алгоритм 1.

Шаг 1. Присвоить $j = 1$.

Шаг 2. Произвести вычисления по формуле $a_{jj} = a_{jj} - \sum_{i=s_j}^{j-1} a_{ji} \cdot a_{ji} \cdot a_{ii}$.

Шаг 3. Для всех $i \in T_j$, если $s_i \le j$, то $a_{ij} = \left(a_{ij} - \sum_{k=\max(s_i, s_j)}^{j-1} a_{ik} \cdot a_{jk} \cdot a_{kk}\right) \Big/ a_{jj}$.

Шаг 4. Увеличить j на единицу, если $j < n + 1$, перейти к шагу 2.

Число итераций на шаге 3 алгоритма 1 напрямую зависит от количества номеров строк в T_j. Очевидно, что номера i, для которых $s_i > j$, могут быть исключены из T_j. Некоторые из рассмотренных далее алгоритмов позволяют заранее исключать из T_j эти номера, упрощая алгоритм. Далее будем называть T_j вертикальной структурой матрицы $\mathbf{A}$.

После выполнения алгоритма 1 из элементов, расположенных на главной диагонали матрицы $\mathbf{A}$ и ниже ее, можно сформировать диагональную и треугольные матрицы разложения $\mathbf{LDL}^{\mathrm{T}}$, где

$$\mathbf{D} = \mathrm{diag}(a_{11}, a_{22},\ldots, a_{nn}), \quad \mathbf{L} = (l_{ij}), \quad l_{ij} = \begin{cases} 0, & j > i \\ 1, & j = i \\ a_{ij}, & j < i \end{cases}, \quad i, j = 1,2,\ldots,n.$$

Разложение $\mathbf{LDL}^{\mathrm{T}}$ позволяет разбить решение исходной системы на три этапа: 1) $\mathbf{Lx}' = \mathbf{b}$ (прямая подстановка); 2) $\mathbf{x}'' = \mathbf{D}^{-1}\mathbf{x}'$; 3) $\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{x}''$ (обратная подстановка). Решая систему подобным образом, мы получаем преимущество от симметричности (вдвое меньшее число операций для разложения), положительной определенности (нет необходимости выбирать разрешающий элемент) и разреженности ее матрицы. Чтобы применение данного алгоритма было максимально эффективным, необходимо таким образом упорядочить элементы матрицы $\mathbf{A}$, чтобы в начале каждой строки оказалось как можно больше нулей, т.е., чтобы номера s_i были максимальными.

Другими словами, необходимо минимизировать заполнение матрицы $\mathbf{A}$. Здесь под заполнением будем понимать величину η:

$$\eta = n + 2\sum_{i=1}^{n}(i - s_i).$$

Минимизация заполнения матрицы системы

Задачи, описанные в разделе «Структура типичных задач», сводятся к решению систем линейных алгебраических уравнений с матрицами, состоящими из блоков $\mathbf{B}_{ij}$, где $i, j = 1, 2,\ldots, m$, m – количество изображений в задаче. При этом блок $\mathbf{B}_{ij}$ содержит ненулевые элементы только когда изображения с номерами i и j связаны (в зависимости от конкретной задачи изображения пересекаются или содержат проекции общей точки пространства). Чем ближе номера связанных изображений друг к другу (чем меньше величина $|i–j|$), тем ближе к главной диагонали будут располагаться блоки с ненулевыми элементами и тем меньше будет заполнение η матрицы системы **A**.

Набор изображений и их связи могут быть представлены в виде графа $G(V, E)$, где V – множество номеров изображений (вершины графа), а E – множество связей между этими изображениями (ребра графа). Используя данное представление, рассмотрим три эвристических алгоритма на графах для перенумерации изображений, приводящих к низкому заполнению η матрицы системы.

Обратный алгоритм Катхилла–Макки

Самым известным алгоритмом перенумерации изображений является алгоритм Катхилла–Макки [4]. Для описания этого алгоритма введем понятие структуры уровней **L**. Структура $\mathbf{L} = (\mathbf{l}_1, \mathbf{l}_2,\ldots, \mathbf{l}_n)$ представляет собой вектор размерности n, где n – количество уровней в структуре. Каждый уровень $\mathbf{l}_i = \{l_{i1}, l_{i2},\ldots, l_{i\lambda(i)}\}$, $i \in 1, 2,..., n$ – вектор размерности λ_i, где $\lambda_i = \lambda(i)$ – количество вершин на уровне i. Структура уровней строится на основе вершины $r \in V$ графа изображений $G(V, E)$. Эта вершина целиком заполняет первый уровень структуры **L**: $l_{11} = r$, $\lambda(1) = 1$. Каждый следующий уровень $\mathbf{l}_i$, $i = 2, 3,..., n$ состоит из элементов множества Ψ_i:

$$\Psi_i = \{k \in \bigcup_{j=1}^{\lambda(i-1)} \mathrm{Adj}(l_{i-1,j}) \mid k \notin \Psi_{i-1} \cup \Psi_{i-2}\}.$$

Здесь $\mathrm{Adj}(v)$ – множество всех вершин, смежных с вершиной v, а $\Psi_{i–1}$ и $\Psi_{i–2}$ – множества, составленные из компонент векторов $\mathbf{l}_{i–1}$ и $\mathbf{l}_{i–2}$ соответственно. Множество $\Psi_0 = \varnothing$. Порядок компонент в векторе $\mathbf{l}_i$ для произвольной структуры уровней может быть любым. Однако в алгоритме 2 будет выбран определенный порядок, который позволит упростить алгоритм Катхилла–Макки.

Построение структуры уровней для некоторой вершины r легко реализовать, если хранить граф $G(V, E)$ в виде двух векторов: $\mathbf{d} = (d_1, d_2,\ldots, d_{|V|})$ и $\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2,\ldots, \mathbf{c}_{|V|})$. Каждый элемент d_i хранит количество вершин, связанных с вершиной i, а вектор $\mathbf{c}_i = (c_{i1}, c_{i2},\ldots, c_{id(i)})$ хранит номера вершин, связанных с вершиной i в порядке возрастания их степени (количества ребер графа, инцидентных этой вершине), $i \in 1, 2, ..., |V|$, $d(i) = d_i$. Подобная структура может быть построена из набора ребер E всего за один проход по ним. Хранение связанных элементов в порядке возрастания их степени необходимо для реализации алгоритма Катхилла–Макки. Используя описанную структуру, приведем алгоритм построения структуры уровней **L** на основе вершины r:

Алгоритм 2.

Шаг 1. Инициализировать: $l_{11} = r$, $\lambda_1 = 1$, $f_i = 0$, $i = 1, 2,\ldots, |V|$, $i \neq r$, $f_r = 1$, $n = 1$.

Шаг 2. Присвоить: $j = 1$, $i = 1$.

Шаг 3. Присвоить $p = l_{ni}$.

Шаг 4. Для $k = 1, 2,\ldots, d_p$: если $f_{c(p,k)} = 0$, присвоить: $f_{c(p,k)} = 1$, $l_{n+1,j} = c_{pk}$, $j=j+1$.

Шаг 5. Увеличить i на единицу, если $i \leq \lambda_n$, перейти к шагу 3.

Шаг 6. Присвоить: $\lambda_{n+1} = j-1$, если $\lambda_{n+1}>0$, то $n = n + 1$ и перейти к шагу 2.

Используя структуру уровней, полученную с помощью алгоритма 2, запишем обратный алгоритм Катхилла–Макки [4] следующим образом:

Алгоритм 3.

Шаг 1. Присвоить: $p = 1$, $i = n$.

Шаг 2. Для $j = \lambda_i, \lambda_{i-1},\ldots, \lambda_1$ присвоить: $k_{l(i,j)} = p$, $p = p+1$.

Шаг 3. Уменьшить i на единицу, если $i \geq 1$, перейти к шагу 1.

Элементы k_i, $i = 1, 2,\ldots, |V|$ – новые номера вершин в V.

В качестве вершины r для алгоритма Катхилла–Макки необходимо взять вершину, дающую наиболее длинную и узкую структуру уровней. Такая вершина, как правило, находится на границе графа. Назовем ее псевдопереферийной вершиной. Структура уровней $\mathbf{L}$, а также число уровней n зависят от начальной вершины x, для которой данная структура была построена. Рассматривая $\mathbf{L}$ и n как функции вершины x, будем писать: $\mathbf{L}(x)$ и $n(x)$. Псевдопереферийную вершину r можно найти с помощью алгоритма 4.

Алгоритм 4.

Шаг 1. Выбирать в качестве начальной вершины произвольную вершину x.

Шаг 2. Построить структуру уровней $\mathbf{L}(x)$.

Шаг 3. Выбрать из $\mathbf{l}_{n(x)}(x)$ узел r с минимальной степенью.

Шаг 4. Построить структуру уровней $\mathbf{L}(r)$.

Шаг 5. Если $n(r) > n(x)$: $x = r$, перейти к шагу 3.

Алгоритм Катхилла–Макки, кроме минимизации заполнения η, также уменьшает величину w:

$$w = \max(i - s_i + 1), \quad i = 1, 2,..., n,$$

где n – число уравнений в решаемой системе. Использование w в алгоритме 1 для вычисления T_j ($T_j = \{j+1, j+2,..., \min(j+w-1, n)\}$) дает существенный прирост производительности за счет уменьшения числа итераций на шаге 3 этого алгоритма.

Метод вложенных сечений

Метод вложенных сечений [5] основан на рекурсивном разделении графа $G(V, E)$ на несвязные компоненты с помощью вершин-разделителей. Алгоритм этого метода можно записать следующим образом:

Алгоритм 5.

Шаг 1. Присвоить $R = V$, $k = |V|$.

Шаг 2. Выбрать в $G(R, E)$ связную компоненту $G(C, E)$, построить для нее структуру уровней $\mathbf{L}(r) = \{\mathbf{l}_1, \mathbf{l}_2,\ldots,\mathbf{l}_n\}$, где r – псевдопериферийная вершина.

Шаг 3. Если в $\mathbf{L}(r)$ число уровней $n(r) \leq 3$, положить $S = C$ и перейти к шагу 4. Иначе $j = \lceil n/2 \rceil$, $S = \{y \in \mathbf{l}_j | \text{Adj}(y) \cap \mathbf{l}_{j+1} \neq \varnothing\}$. Здесь $\lceil x \rceil = \min\{n \in \mathbb{Z} | n \geq x\}$.

Шаг 4. Пронумеровать узлы разделителя S числами от $k - |S| + 1$ до k.

Шаг 5. Присвоить: $R = R - S$ и $k = k - |S|$. Если $R \neq \varnothing$, перейти к шагу 2.

На шаге 2 алгоритма 5 происходит выбор связной компоненты из графа $G(R, E)$. Наилучшим решением является наименьшая по размеру компонента. Такая стратегия выбора позволяет уменьшить разброс элементов вокруг главной диагонали для строк и столбцов матрицы решаемой системы уравнений, соответствующих длинным разделителям.

Алгоритм вложенных сечений в отличие от алгоритма Катхилла–Макки не минимизирует величину w. Однако при многократном решении задачи с одинаковой структурой можно рассчитать вертикальную структуру матрицы $T(j)$ следующим образом: $T(j) = \{y \in \{j + 1, j + 2, ..., n\} | \mathbf{s}_y \leq j\}$. Использование заранее рассчитанной структуры $T(j)$ существенно упрощает последующее решение задач с однотипной вертикальной структурой с помощью алгоритма 1.

Модификация метода вложенных сечений

Наиболее важный шаг в алгоритме 5 – поиск разделителя S. Разделитель на каждой итерации должен состоять из как можно меньшего числа вершин и делить граф на сопоставимые по размеру части. В алгоритме 5 размер разделителя никак не учитывается. В качестве разделителя на шаге 3 берутся такие элементы центрального уровня j, которые имеют связи с элементами нижнего уровня. Данная стратегия выбора очень проста, но не эффективна. Ее можно улучшить, если выбирать в качестве уровня j не центральный, а минимальный уровень из некоторой окрестности центра структуры уровней.

Рассмотрим следующую окрестность уровня j: $\lceil n/2 - an \rceil$, $\lceil n/2 - an + 1 \rceil$,..., $\lceil n/2 + an \rceil$, где a – коэффициент от 0 до 0.5. Из данной окрестности выбирается такой уровень j, который дает разделитель S минимального размера. При $a = 0$ имеем алгоритм 5. При $a = 0.5$ имеем ситуацию, когда на каждой итерации выбирается минимальный уровень из всех уровней в структуре. Однако подобная ситуация приводит к неравномерному делению графа и существенно снижает скорость работы алгоритма.

На рис. 1 представлены графики зависимости полученного заполнения матрицы от коэффициента a для двух разных задач. Выигрыш, получаемый при выборе уровня разделителя из некоторого интервала, существенно зависит от характера задачи.

Например, на левом графике рис. 1 при $a = 0.35$ заполнение оказалось в 1.4 раза меньше, чем при $a=0$, а на правом графике при $a = 0.35$ заполнение уменьшилось в 3 раза.

Для сравнения алгоритмов в последующих разделах используются две версии алгоритма вложенных сечений: версия с коэффициентом $a=0.35$ и версия, выполняющая алгоритм несколько раз для различных коэффициентов a и выбирающая минимальное получившееся заполнение.

Следует отметить, что модифицированный алгоритм вложенных сечений рассматривается в контексте задач компьютерного зрения, которые характеризу-

ются непостоянным количеством связей между изображениями, что порождает разные по ширине уровни в структуре уровней.

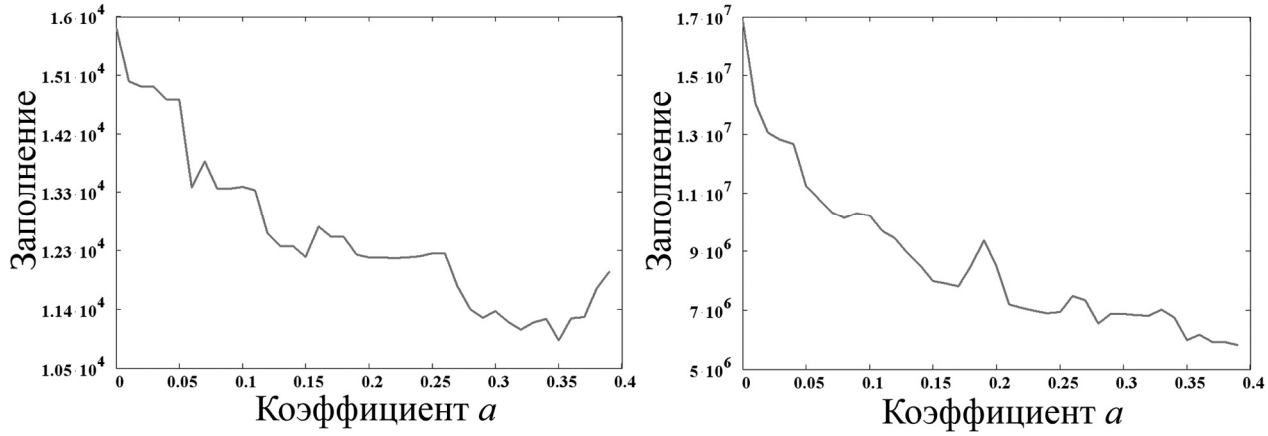

Рис. 1. Зависимость матрицы системы от коэффициента *a*.

Для задач на регулярных сетках модифицированный алгоритм приводит лишь к увеличению времени вычислений, так как практически все уровни будут иметь одинаковую ширину.

Метод параллельных сечений

Метод параллельных сечений заключается в единовременном разделении графа $G(V, E)$ на q компонент связности R_i, $i = 1, 2,\ldots, q$ с помощью разделителей $S_j, j = 1, 2,\ldots, q - 1$. Структура уровней **L**, построенная для графа $G(V, E)$, делится с помощью $q - 1$ горизонтальных параллельных разделителей S_j, как это описано в [5]. При этом для компонент связности и разделителей справедливы соотношения:

$$R_i \cap R_k = \varnothing,\ k = 1, 2,\ldots, q,\ k \neq i,$$
$$S_j \cap S_p = \varnothing,\ p = 1, 2,\ldots, q-1,\ |j - p| > 1. \qquad (2)$$

Нумерация узлов в методе параллельных сечений происходит следующим образом: узлы всех компонент R_i нумеруются отдельно с помощью алгоритма Катхилла–Макки, затем к номеру добавляется число $\sum_{k=1}^{i-1}|R_k|$. Нумерация внутри разделителей S_j произвольна, а для получения глобального номера узла к внутреннему номеру добавляется число $\sum_{i=1}^{q}|R_i| + \sum_{k=1}^{j-1} S_k$. Подобная нумерация узлов приводит к матрице системы, имеющей вид, представленный на рис. 2 слева. Эта матрица состоит из блоков $\mathbf{B}_{kp}$, $k, p = 1, 2,\ldots, 2q - 1$. Ширина блока $\mathbf{B}_{kp}$ равна $|R_p|$, при $p \le q$ и $|S_{p-q}|$ – при $p > q$. Высота блока $\mathbf{B}_{kp}$ равна $|R_k|$, при $k \le q$ и $|S_{k-q}|$ – при $k > q$.

На основе (2) можно сделать вывод, что блок $\mathbf{B}_{kp}$ содержит ненулевые элементы только в трех случаях: $k = p$, $0 \le k - q - p \le 1$ или $0 \le p - q - k \le 1$ (серые блоки на рис. 2).

Подобная структура матрицы не дает минимального заполнения η, но сохраняет большое количество дополнительных пустых блоков после $\mathbf{LDL}^{\mathrm{T}}$ разложения.

На рис. 2 справа показана схема матрицы **L**. Черные блоки – блоки, которые стали ненулевыми после $\mathbf{LDL}^{\mathrm{T}}$ разложения.

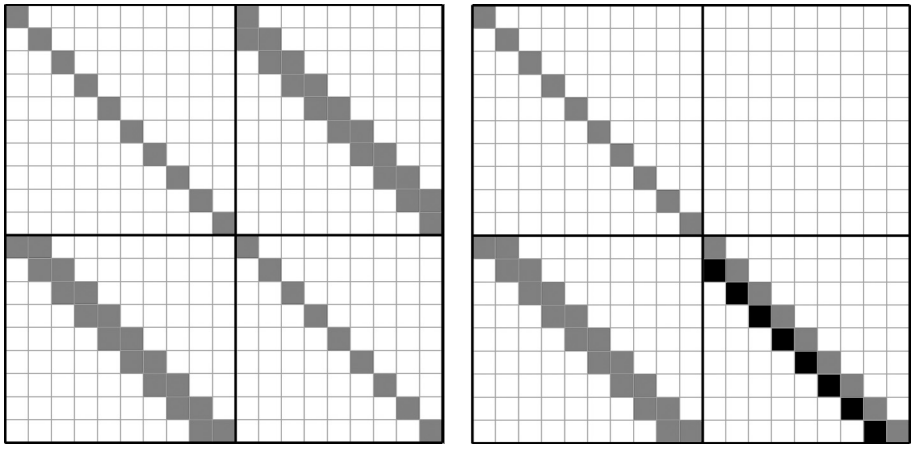

Рис. 2. Метод вложенных сечений: слева – структура матрицы системы; справа – структура матрицы L из LDL^T разложения.

Кроме блоков с индексами k, p: $p < k$, нулевыми остались дополнительные блоки с индексами i, j: $i > q$, $i - q + 1 < j < i - 1$, а также блок с индексами $q + 1$, q. Если пропускать дополнительные пустые блоки в алгоритме 1, можно существенно уменьшить объем вычислений. Таким образом, говоря о заполнении матрицы системы в случае метода параллельных сечений, будем иметь в виду величину η', учитывающую дополнительные пустые блоки:

$$\eta' = \eta - 2\sum_{i=1}^{q-1}|S_i|\left(\sum_{j=i+2}^{q}|R_j| + \sum_{j=1}^{i-2}|S_j|\right).$$

Исходные данные для сравнения алгоритмов

В качестве тестовой задачи для алгоритмов выберем описанную линейную задачу МНК. К упорядочиванию изображений в данной задаче применим четыре алгоритма: обратный алгоритм Катхилла–Макки (ОКМ), модифицированный алгоритм вложенных сечений $a = 0.35$ (ВС1), модифицированный алгоритм вложенных сечений для диапазона a (ВС2), алгоритм параллельных сечений (ПС).

Возьмем 6 различных областей, покрытых изображениями. Две области, показанные на рис. 3а,б – фотографии морского дна, полученные с автономного необитаемого подводного аппарата (АНПА). Другие 4 области представляют синтетические данные, сгенерированные для проверки эффективности рассматриваемых алгоритмов в различных ситуациях, которые могут возникнуть на практике (рис. 3в-е).

Табл. 1 содержит подробное описание используемых для тестирования наборов изображений. Каждая из четырех представленных на рис. 3 синтетических областей формирует четыре задачи. Два вида задач – lo и hi – характеризуются числом изображений m. Для задач типа lo оно низкое: 23000÷25000, а для задач hi – высокое: 90000÷100000. Задачи типа lo используются для тестирования алгоритмов упорядочения и решения систем уравнений. Задачи типа hi используются лишь для тестирования эффективности алгоритмов упорядочения.

Другие два вида задач – weak и strong. Задачи weak характеризуются малым числом n связей между изображениями. Отношение числа связей к числу изображений в weak-задачах не превышает 3.6.

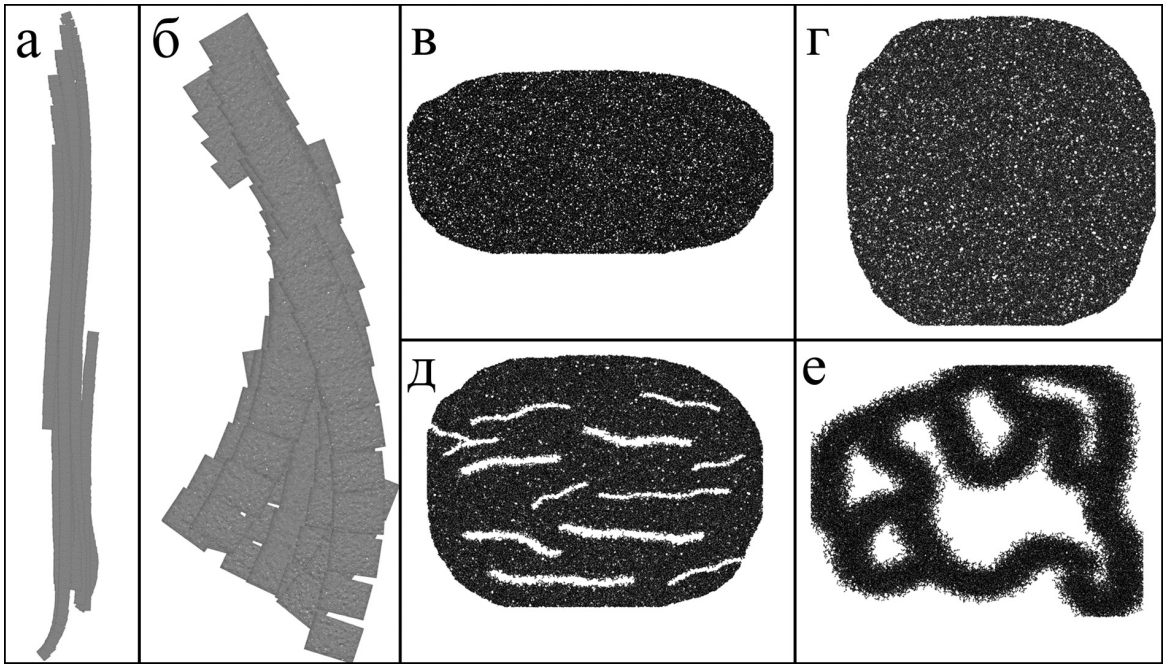

Рис. 3. Набор областей, покрытых изображениями.

Таблица 1

Название	Описание	*m*	*n*	Рисунок
R1	Сходящийся пучок траекторий АНПА	90	216	Рис 3, а
R2	Вытянутый участок морского дна	647	1904	Рис 3, б
S1_weak_lo	Вытянутый участок территории, полностью покрытый изображениями	24209	62641	Рис 3, в
S1_weak_hi		91748	219517	
S1_strong_lo		25000	252201	
S1_strong_hi		100000	920871	
S2_weak_lo	Округлый участок территории, полностью покрытый изображениями	24266	64268	Рис 3, г
S2_weak_hi		97360	257078	
S2_strong_lo		25000	257576	
S2_strong_hi		100000	1038573	
S3_weak_lo	Участок территории с маленькими зонами, не покрытыми фотографиями	23749	67968	Рис 6, д
S3_weak_hi		92273	233430	
S3_strong_lo		25000	274273	
S3_strong_hi		99979	1228105	
S4_weak_lo	Участок территории с большими зонами, не покрытыми фотографиями	23477	82797	Рис 6, е
S4_weak_hi		91938	329076	
S4_strong_lo		24911	331963	
S4_strong_hi		99390	1335892	

В общем случае weak-задача соответствует идеальному расположению изображений: они имеют достаточное для решения задачи, но не избыточное перекрытие. Однако на практике часто бывают ситуации, когда одна и та же зона покрывается фотографиями большое количество раз. Например, в задаче трехмерной реконструкции по изображениям [2]. В этом случае рассматриваются strong-задачи. В них отношение числа связей к числу изображений достигает 13.5. Таким образом, для каждой синтетической области рассматриваются четыре задачи: weak_lo, weak_hi, strong_lo и strong_hi.

Сравнение алгоритмов

При тестировании все алгоритмы выполнялись на одном ядре процессора Intel Core2 Quad Q9550 2.83GHz. В качестве основного критерия эффективности рассмотренных алгоритмов было взято общее время решения системы уравнений. Табл. 2 отображает время решения системы t_s для каждой из описанных областей каждым алгоритмом, а также время упорядочения t_o.

Таблица 2

Название	ОКМ		ВС1		ВС2		ПС	
	t_o, с	t_s, с	t_o, с	t_s, с	t_o, с	t_s, с	t_o, с	t_s, с
R1	**0.001**	**0.010**	0.001	0.015	0.001	0.013	0.001	0.016
R2	**0.001**	**0.109**	0.001	0.260	0.047	0.250	0.001	0.150
S1_weak_lo	**0.014**	**3.984**	0.156	8.078	5.063	5.469	0.016	11.422
S1_weak_hi	**0.032**	-	1.079	-	30.235	-	0.062	-
S1_strong_lo	**0.015**	**32.140**	0.313	198.485	7.953	152.593	0.032	148.000
S1_strong_hi	**0.047**	-	1.843	-	61.844	-	0.125	-
S2_weak_lo	**0.015**	**6.797**	0.187	8.641	6.187	8.172	0.015	20.156
S2_weak_hi	**0.032**	-	1.079	-	30.235	-	0.062	-
S2_strong_lo	**0.015**	**65.125**	0.296	198.765	7.969	185.937	0.016	197.109
S2_strong_hi	**0.078**	-	2.047	-	58.250	-	0.140	-
S3_weak_lo	**0.016**	**4.256**	0.141	5.922	4.297	5.907	0.015	11.688
S3_weak_hi	**0.031**	-	0.812	-	27.016	-	0.047	-
S3_strong_lo	**0.016**	**49.391**	0.281	85.844	6.812	77.718	0.016	151.812
S3_strong_hi	**0.078**	-	2.063	-	81.015	-	0.140	-
S4_weak_lo	**0.015**	**4.406**	0.125	7.109	3.359	7.094	0.015	11.485
S4_weak_hi	**0.047**	-	0.922	-	26.781	-	0.062	-
S4_strong_lo	**0.015**	**40.781**	0.281	82.454	7.484	76.656	0.016	139.375
S4_strong_hi	**0.078**	-	2.078	-	55.688	-	0.156	-

Из табл. 2 видно, что для всех задач обратный алгоритм Катхилла–Макки дал наилучшие результаты как по времени решения системы, так и по времени упорядочения. Отчасти это объясняется тем, что данный алгоритм не только минимизирует заполнение, но и дает малую ширину диагонали w, что упрощает шаг 3 алгоритма 1.

Очевидно, что при наличии информации о вертикальной структуре матрицы системы T обратный алгоритм Катхилла–Макки лишится данного преимущества. Поэтому в качестве еще одного критерия оценки эффективности описанных алгоритмов приведем заполнение матрицы системы после упорядочения (табл. 3). Для удобства в табл. 3 для каждой задачи занесено не само заполнение η (η' для алгоритма ПС), а величина $f = \eta/(4m)$ ($f = \eta'/(4m)$ для алгоритма ПС). Эта величина характеризует среднее расстояние между первым ненулевым элементом каждой строки и главной диагональю. Кроме информации о заполнении, в табл. 3 присутствует информация об оперативной памяти M, требующейся для решения каждой задачи каждым алгоритмом. Из табл. 3 видно, что выигрывающий по времени обратный алгоритм Катхилла–Макки далеко не всегда дает наилучшее заполнение. В большинстве weak-задач модифицированные алгоритмы вложенных сечений ВС1 и ВС2 дали лучшие результаты. Однако на strong-задачах, а также на задачах R1 и

R2, для которых характерна очень узкая структура уровней, обратный алгоритм Катхилла–Макки по-прежнему лидирует.

Таблица 3

Название	ОКМ		ВС1		ВС2		ПС	
	M, Мб	*f*	*M*, Мб	*f*	*M*, Мб	*f*	*M*, Мб	*f*
R1	**0.011**	**8**	0.021	16	0.019	14	0.020	14
R2	**0.156**	**16**	0.347	32	0.333	34	0.286	30
S1_weak_lo	52.126	142	66.079	178	**42.506**	**116**	86.137	232
S1_weak_hi	334.645	240	250.725	180	**171.547**	**122**	510.300	364
S1_strong_lo	**196.091**	**514**	468.369	1228	413.666	1084	392.806	1030
S1_strong_hi	**1460.042**	**956**	3265.106	2140	3032.128	1988	2550.767	1672
S2_weak_lo	77.374	208	67.960	184	**63.648**	**172**	124.098	336
S2_weak_hi	612.313	412	569.415	384	**519.744**	**350**	920.646	620
S2_strong_lo	**277.973**	**726**	478.717	1256	462.212	1212	449.822	1180
S2_strong_hi	**2245.680**	**1472**	3740.140	2452	3645.476	2390	3803.769	2492
S3_weak_lo	53.235	148	45.792	126	**44.251**	**122**	83.664	230
S3_weak_hi	322.112	228	210.117	150	**177.527**	**126**	513.800	366
S3_strong_lo	**241.962**	**634**	309.085	810	296.054	776	396.088	1038
S3_strong_hi	**2036.948**	**1336**	3242.107	2126	2713.446	1778	3297.019	2162
S4_weak_lo	**56.105**	**156**	58.282	162	58.212	162	87.508	244
S4_weak_hi	**433.399**	**310**	523.666	374	475.545	340	675.012	482
S4_strong_lo	**212.899**	**560**	300.517	790	290.664	764	375.373	986
S4_strong_hi	**1976.542**	**1304**	2555.019	1684	2415.144	1592	3526.447	2326

Заключение

На основе исследования эффективности описанных алгоритмов для задач компьютерного зрения, сводящихся к решению линейной и нелинейной задач МНК, можно сделать следующие выводы. Алгоритм параллельных сечений не оказался лучшим ни для одной из рассматриваемых областей. Кроме того, данному алгоритму необходима особая структуру данных для хранения матрицы системы, а также модификация алгоритма 1, учитывающая эту структуру.

Алгоритм вложенных сечений в исходном варианте [5] также не показал хороших результатов (информация о нем не внесена в табл. 2 и 3), однако модифицированные алгоритмы вложенных сечений, предложенные в данной работе, оказались самыми эффективными при минимизации заполнения в задачах, где изображения слабо связаны. Данные алгоритмы можно рекомендовать при необходимости многократного решения систем с одинаковой структурой, – например, в нелинейных задачах МНК. Вертикальная структура T в таком случае может быть рассчитана один раз и использована для решения всех систем. Это дает линейную зависимость времени решения систем от заполнения η.

Наиболее подходящим для задач компьютерного зрения показал себя обратный алгоритм Катхилла–Макки. Им следует пользоваться, когда задача не предполагает многократного решения систем уравнений с одинаковой структурой, – например, в линейных задачах МНК. Также данный алгоритм дает наилучшие результаты в задачах с очень плотным покрытием изображениями и в задачах, где область, покрываемая изображениями, узкая.

Применение алгоритма Катхилла–Макки или модифицированных алгоритмов вложенных сечений совместно с методом решения систем, учитывающим разреженность, позволяет за разумное время решать на современном персональном компьютере задачи компьютерного зрения с десятками тысяч связанных изображений.

ЛИТЕРАТУРА

1. *Marquard D.* An algorithm for least-squares estimation of nonlinear parameters // SIAM Journal on Applied Mathematics. – 1963. – Vol. 11, №2. – P. 431-441.
2. *Hartley R., Zisserman A.* Multiple View Geometry in Computer Vision. – Cambridge University Press, 2004.
3. *Golub G.H., Van Loan C.F.* Matrix Computations. – The Johns Hopkins University Press, Baltimore, MD, second edition, 1989.
4. *Cuthill E., McKee J.* Reducing the bandwidth of sparse symmetric matrices // Proc. of the 24-th National Conference ACM. – New York, 1969. – P. 157-172.
5. *Джордж А., Лю Дж.* Численное решение больших разреженных систем уравнений / пер. с англ. – М.: Мир, 1984.
6. *Karypis G., Kumar V.* A fast high quality multilevel scheme for partitioning irregular graphs // SIAM Journal on Scientific Computing. – 1999. – Vol. 20, №1. – P. 359-392.

Статья представлена к публикации членом редколлегии С.И. Смагиным.

E-mail:
Камаев Александр Николаевич – kamaev_an@mail.ru.

УДК: 004.94+51-73

© 2013 г. **А.Г. Масловская,** канд. физ.-мат. наук
(Амурский государственный университет, Благовещенск)

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ФОРМИРОВАНИЯ КОНТРАСТА РЭМ-ИЗОБРАЖЕНИЙ ДОМЕННЫХ СТРУКТУР СЕГНЕТОЭЛЕКТРИКОВ

Предложена система имитационного моделирования изображений сегнетоэлектрических доменных структур, наблюдаемых с помощью методик растровой электронной микроскопии. Алгоритм основан на математической модели формирования видеосигнала в пироэлектрическом режиме. Представлены результаты вычислительного эксперимента.
Ключевые слова: математическая модель, алгоритм, компьютерное моделирование, доменная структура, сегнетоэлектрик, растровое изображение.

Введение

В настоящее время применение методов математического и компьютерного моделирования является одним из важнейших составляющих в решении фунда-