УДК 004.932.2

И. В. Тупицын

БЫСТРЫЙ АЛГОРИТМ РЕКОНСТРУКЦИИ ПРОМЕЖУТОЧНЫХ РАКУРСОВ ПО СТЕРЕОПАРЕ

Рассмотрен эффективный алгоритм стереореконструкции А. А. Лукьяницы. Предложена модификация данного алгоритма, позволяющая увеличить скорость его работы. Представлены результаты проведенных экспериментов.

Ключевые слова: стереоизображение, эпиполярная геометрия, соответствующие точки, диспарантность.

Одно из современных направлений представления видеоинформации связано с разработкой устройств, способных воспроизводить объемные изображения (многоракурсные дисплеи, проекционные системы). Для работы этих устройств необходимо получить заданное количество ракурсов воспроизводимой сцены, количество которых определяется конкретной задачей. Например, для комфортного просмотра видео бывает достаточно 4…6 ракурсов, тогда как для таких применений, как 3*D*-томография и рентгеноскопия, графические рабочие станции CAD/CAM, отображение оперативной обстановки для авиадиспетчерских, аварийно-спасательных служб и т. д., может понадобиться от 40 до 150 ракурсов. Известны специализированные устройства, обеспечивающие от двух (стереовидеокамеры) до восьми ракурсов сцены. Восьмиракурсная видеокамера применялась в прототипе многоракурсной телевизионной системы НИКФИ, а видеокамеру с бульшим числом ракурсов пока невозможно представить. Не менее проблематичной является запись и передача по каналам связи такого сигнала. Следовательно, возникает потребность в реконструкции необходимого числа промежуточных ракурсов, т. е. изображений, которые были бы получены аналогичной камерой, если бы она занимала промежуточные положения между камерами, используемыми при стереосъемке.

Большинство алгоритмов решают эту задачу в два этапа: первый этап – поиск соответствующих точек, т. е. точек на изображениях, соответствующих одной и той же области трехмерного пространства; второй этап – аппроксимация промежуточных ракурсов по найденным точкам. Как правило, если найдены все соответствующие точки, то аппроксимация промежуточных ракурсов не вызывает затруднений, поэтому основной задачей при реконструкции промежуточных ракурсов является поиск соответствующих точек.

В качестве примера рассмотрим работу простого алгоритма стереозрения (рис. 1). Он состоит из нескольких этапов:

1) в качестве текущей линии выбирается первая (верхняя) эпиполярная линия;

2) для текущей эпиполярной линии в качестве текущего выбирается самый левый пиксель;

3) текущий пиксель левого изображения сопоставляется с каждым пикселем правого изображения;

4) выбирается наиболее похожий по некоторой мере пиксель из правого изображения и записывается в массив соответствующих пикселей. Этот массив представляет собой двумерный целочисленный массив, индексами элементов которого являются позиции пикселей левого изображения, а значениями элементов – позиции соответствующих им пикселей правого изображения;

5) если текущий пиксель не последний в текущей строке, то следующий пиксель становится текущим пикселем и алгоритм переходит к п. 3. В противном случае переход к п. 6;

6) если текущая эпиполярная линия не последняя, то следующая эпиполярная линия становится текущей и алгоритм переходит к п. 2. В противном случае работа алгоритма завершается.

Результатом работы данного алгоритма является массив соответствующих пикселей. Используя этот массив, можно построить карту глубины или реконструировать промежуточные ракурсы.

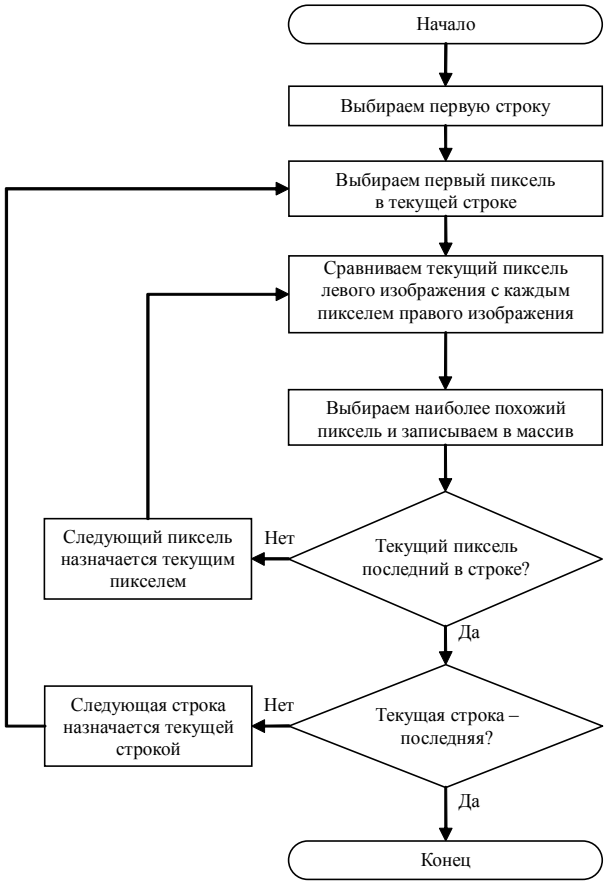


Рис. 1. Блок-схема работы простого алгоритма стереозрения

Реконструкция промежуточных ракурсов по методу А. А. Лукьяницы. Постановка задачи. Пусть нам дана стереопара, т. е. два изображения одной и той же сцены,

зафиксированные стереокамерой. Обозначим изображение, полученное левой камерой, буквой L, а изображение правой камеры – буквой R. Введем ортогональную систему координат, согласованную с камерами: ось x направим вправо через центры изображений, ось y – вниз вдоль вертикальной оси изображений, ось z дополнит их до правой тройки. Начало координат выберем таким образом, чтобы изображения находились в плоскости Oxy, а центры изображений были симметричны относительно плоскости Oyz (рис. 2). Сцена зарегистрирована двумя камерами: L и R. Следует построить промежуточный ракурс M.

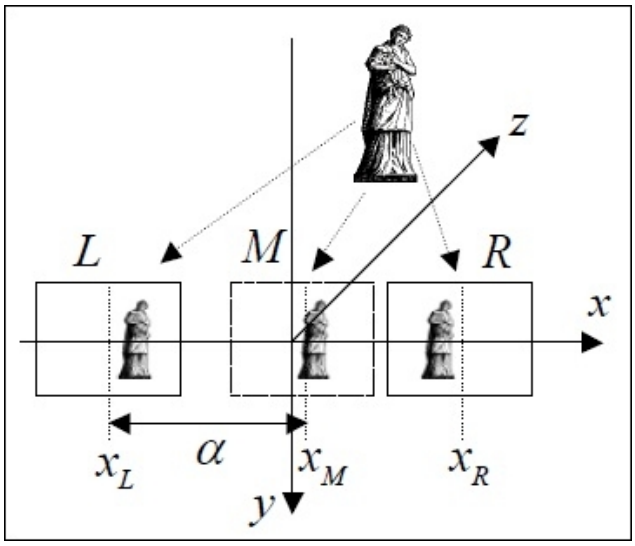


Рис. 2. Геометрия стереосъемки

Пусть центр левого изображения в выбранной системе координат имеет координаты $(x_L, 0, 0)$, а центр правого – $(x_R, 0, 0)$. Задача состоит в том, чтобы построить изображение в плоскости Oxy с центром в некоторой точке $(x_M, 0, 0)$, наиболее близкое к изображению, которое было бы получено камерой, оптическая ось которой проходит через точку $(x_M, 0, 0)$.

Метод А. А. Лукьяницы реализуется в два этапа [1]:

1) находятся сопряженные точки на левом и правом изображениях;

2) промежуточное изображение аппроксимируется по сопряженным точкам и заполняются области, для которых соответствие не является однозначным.

Алгоритм поиска сопряженных точек. Пусть каждое изображение содержит $K \times N$ пикселей по горизонтали и вертикали соответственно. Задача нахождения сопряженных точек состоит в следующем: для каждого пикселя левого изображения L_{ki}, $k = 1, \ldots, K$, $i = 1, \ldots, N$, следует найти пиксель R_{kj}, $j = 1, \ldots, N$, на правом изображении, соответствующий той же точке регистрируемой сцены. Предположим, что камеры установлены на одной горизонтали и имеют одинаковый коэффициент усиления. Тогда горизонтальные строки фотоприемников совпадают с эпиполярными линиями, поэтому сопряженные точки должны находиться на строках, имеющих одинаковый номер. Это позволяет записать следующее выражение для вычисления эпиполярных линий:

$$L(k, i) \approx R(k, j + \Delta_j),$$

где Δ_j – смещение, зависящее от номера пикселя.

Нахождение сопряженных точек состоит в вычислении смещений для всех пикселей. Далее для простоты будем опускать первый индекс, поскольку он имеет одно и то же значение для сравниваемых изображений, и подразумевать, что описываемый ниже процесс выполняется для каждой пары соответствующих строк на левом и правом изображениях.

Для нахождения подходящих кандидатов в сопряженные пиксели необходимо сравнивать области Ω в окрестности этих пикселей, что существенно повышает устойчивость алгоритма. Действительно, в изображениях одного и того же участка сцены под различными ракурсами могут появиться отличия, связанные со взаимным расположением источников света и элементов сцены. Кроме того, некоторые области могут быть видимы для одной камеры и невидимы для другой.

Пусть сравниваемая область Ω имеет размеры $k \times n$. Тогда в окрестности j-го пикселя на правом изображении следует просканировать область большего размера $m \times l$, $m \geq k$, $l > n$. В процессе сканирования вычисляется функция расстояния $d(i, j)$, характеризующая близость изображений, находящихся в окрестности Ω_i пикселя $L(i)$ и в окрестности Ω_j пикселя $R(j)$. В качестве функции расстояния обычно выбирается сумма разностей интенсивностей соответствующих пикселей из окресностей Ω_i и Ω_j:

$$d(i, j) = \sum_{m=-k/n}^{k/2} \sum_{l=-n/2}^{n/2} \left| L(k + m, i + l) - R(k + m, j + l) \right|,$$

или сравниваются градиенты изображений из Ω_i и Ω_j.

На практике лучше использовать сочетание этих методов, поскольку они имеют различную эффективность для разных типов изображений. Наиболее часто применяется метод сравнения интенсивностей, но при обработке изображений с однородными пространными объектами, как, например, стена дома, этот метод приводит к появлению плато у функции $d(i, j)$, что затрудняет выбор соответствующих точек. Метод сравнения разности градиентов в этом случае позволяет учесть более тонкие детали изображений.

Для выбора соответствующих точек из всех возможных претендентов нужно, чтобы были удовлетворены два условия. Во-первых, очевидно, что номера индексов соответствующих точек должны минимизировать суммарные отличия сравниваемых областей на левом и правом изображениях. Во-вторых, величина смещения не может уменьшаться по мере возрастания индекса j, т. е. $\Delta_j \geq \Delta_{j-1}$. Это условие следует из геометрических свойств стереопары. Для удовлетворения этих условий предлагается следующий алгоритм, основанный на методе динамического программирования.

Введем две вспомогательные функции: вещественную ψ и целочисленную P. Первая из них нужна для накопления суммарных отличий сравниваемых областей, вторая предназначена для хранения соответствующих индексов.

Алгоритм динамического программирования (рис. 3) состоит из трех этапов:

– инициализации:

$$\psi(i, 1) = d(i, 1), \quad i = 1, \ldots, N,$$
$$P(i, 1) = i, \quad i = 1, \ldots, N;$$

– индуктивного перехода:

$$\psi(i,j) = \min_{1\le k\le i}\psi(k, j-1) + d(i,j),$$
$$i = 2, ..., N, \quad j = 1, ..., N,$$
$$P(i,j) = \arg\min_{1\le k\le i}\psi(k, j-1),$$
$$i = 2, ..., N, \quad j = 1, ..., N;$$

– обратного хода:

$$J(N) = \arg\min_{1\le i\le N}\psi(i, N),$$
$$J(j) = P(J(j+1), j+1),$$
$$j = N-1, ..., 1.$$

В результате будет построен массив индексов для сопряженных точек J, по которому для любого пикселя левого изображения $L(i)$ сопряженный пиксель правого изображения определяется как $R(J(i))$.

Рассмотрим каждый шаг представленного алгоритма подробнее.

Процедура инициализации предназначена для задания начальных значений функций ш и P. Функция ш вычисляется как функция d между первым пикселем левого изображения в текущей строке и каждым из пикселей правого изображения в текущей строке, функция P – как индексы пикселей правого изображения.

Следующий этап алгоритма – индуктивный переход. На этом этапе вычисляются последующие значения функций ψ и P. Функция ψ накапливает суммарные значения сравниваемых областей, что повышает точность поиска сопряженных точек.

Завершающий этап алгоритма динамического программирования – обратный ход. На этом этапе заполняется массив сопряженных пикселей J. В качестве сопряженных для пикселей левого изображения выбираются те пиксели правого изображения, различия между которыми минимальны.

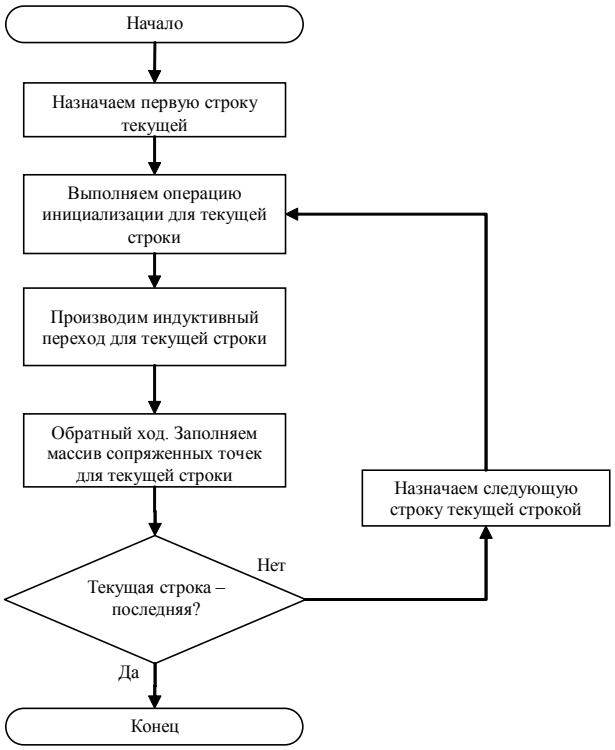


Рис. 3. Блок-схема работы алгоритма динамического программирования

Алгоритм динамического программирования производит обработку изображения построчно, что позволяет распараллелить вычисления. Это значительно ускорит процедуру стереореконструкции, однако следствием построчной обработки может стать эффект «гребенки», с которым данный алгоритм, к сожалению, не борется.

Аппроксимация промежуточных ракурсов. Введем параметр $\alpha = \dfrac{x_M - x_L}{x_M - x_R}$, который характеризует относительную степень близости восстанавливаемого ракурса к левому изображению. Далее для всех значений $i = 1, \ldots, N, j = 1, \ldots, N$ вычислим

$$k = [\alpha j + (1-\alpha)J(j)],$$

$$M(i,k) = \alpha L(i,j) + (1-\alpha)R(i, J(j)).$$

Из этих соотношений следует, что некоторые точки промежуточного ракурса могут оказаться незаполненными. Это происходит из-за того, что соответствующие фрагменты объекта трехмерной сцены видимы одной камерой и невидимы другой.

Пусть в реконструируемом изображении в какой-либо i-й строке оказались незаполненными идущие подряд пиксели с номерами $j_0, \ldots, j_0 + m$. Здесь возможны два случая:

1) если $J(j_0 - 1) = J(j_0 + m + 1)$, то эти пиксели видимы только левой камерой, поэтому они заполняются соответствующими значениями из левого изображения:

$$M(i,j) = L(i,j),$$
$$j = j_0, \ldots, j_0 + m;$$

2) если $J(j_0 - 1) < J(j_0 + m + 1)$, то эти пиксели видимы только правой камерой, поэтому они заполняются соответствующими значениями из правого изображения:

$$M(i,j) = L(i, J(j)),$$
$$j = j_0, \ldots, j_0 + m.$$

Увеличение скорости работы алгоритма. Хотя скорость работы алгоритма динамического программирования довольно высока, она все же может быть недостаточной для решения некоторых задач. Следовательно, необходимо найти способ увеличения скорости работы этого алгоритма.

Как правило, во всех алгоритмах стереозрения большую часть процессорного времени занимает вычисление функции расстояния для пикселей строки. Для того чтобы минимизировать затрачиваемое на это время, необходимо либо упростить саму функции, либо уменьшить количество пикселей, для которых она вычисляется. Возможен вариант уменьшения размера окна вокруг выбранных пикселей, но он не всегда дает хорошие результаты и применяется не для всех изображений.

Если упрощение функции расстояния не представляется возможным, то необходимо уменьшить количество вычислений этой функции. Усовершенствование рассмотренного выше алгоритма связано с введением дополнительной величины – максимального диспаритета $D_{\max}$, т. е. максимальной разницы между смещением точки трехмерного пространства на левом и правом изображениях. Эта величина задается пользователем для каждой стерео-

пары. Ее значение можно вычислить, если известно расположение камер и расстояние от плоскости стереосъемки до максимально отдаленного объекта сцены.

Перепишем выражения, используемые на этапах инициализации, индуктивного перехода и обратного хода, с учетом параметра максимального диспаритета в следующем виде:

– инициализация:

$$\psi(i,1) = d(i,1), \quad i = 1, ..., N,$$
$$P(i,1) = i, \qquad i = 1, ..., N;$$

– индуктивный переход:

$$\psi(i,j) = \min_{D_{\max} \le k \le i} \psi(k, j-1) + d(i,j),$$
$$i = 2, ..., N, \quad j = 1, ..., N,$$
$$P(i,j) = \arg\min_{D_{\max} \le k \le i} \psi(k, j-1),$$
$$i = 2, ..., N, \quad j = 1, ..., N;$$

– обратный ход:

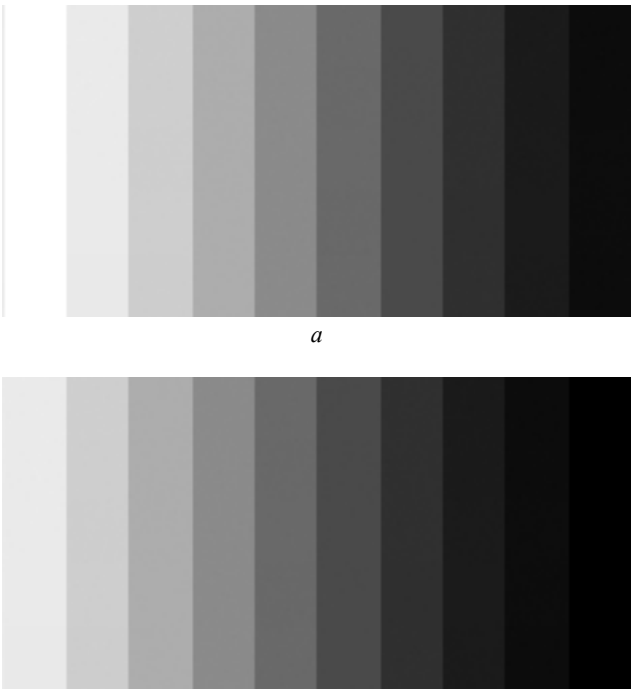
$$J(N) = \arg\min_{D_{\max} \le i \le N} \psi(i,N), \quad J(j) = P(J(j+1), j+1),$$
$$j = N-1, ...,1.$$

В результате введенных изменений скорость работы алгоритма значительно возрастает, поскольку теперь нет необходимости вычислять значение функции расстояний для каждой пары пикселей в строке.

Рассматриваемые в данной статье алгоритмы (А. А. Лукьяницы и модифицированный) были реализованы на языке C++. В ходе проведенных экспериментов измерялось время поиска соответствующих точек одной строки для различных изображений и оценивались различия в результатах работы алгоритмов. Сравнение проводилось на тестовой стереопаре шириной 10 пикселей (рис. 4).

Анализ полученных данных (см. таблицу) показал, что модифицированный алгоритм имеет более высокую скорость работы. Однако при этом показатель количества правильно определенных соответствующих точек ухудшается на 2…6 % по сравнению с базовым алгоритмом А. А. Лукьяницы.

Таким образом, в данной статье был рассмотрен алгоритм стереореконструкции А. А. Лукьяницы и оценена скорость его работы. Предложена модификация этого алгоритма, увеличивающая скорость его работы. Модифицированный алгоритм был реализован на языке программирования высокого уровня C++. По результатам проведенных экспериментов было отмечено возрастание скорости работы усовершенствованного алгоритма при незначительном ухудшении качества стереореконструкции.



а

б

Рис. 4. Тестовая стереопара: *а* – левое изображение; *б* – правое изображение

Библиографическая ссылка

1. Лукьяница А. А. Эффективный алгоритм восстановления промежуточных ракурсов по стереопаре // Graphicon-2006 : материалы Междунар. конф. Новосибирск, 2006. С. 15–18.

Результаты экспериментов

Ширина изображений стереопары	Размер окна сравниваемых областей	Максимальный диспаритет	Выбранный алгоритм	Время работы алгоритма, с
472	5x5	–	Лукьяницы	0,343
		10	Модифицированный	0,047
		20		0,047
716		–	Лукьяницы	1,311
		20	Модифицированный	0,156
		40		0,265
3295		–	Лукьяницы	93,102
		40	Модифицированный	5,304
		100		11,949

I. V. Toupitsyn

FAST ALGORITHM FOR RECONSTRUCTION OF INTERMEDIATE VIEWS FROM STEREOPAIR

An effective algorithm for stereoreconstruction of A. A. Lukianitsa is considered in this article. An algorithm modification, allowing to increase the speed of the algorithm is suggested. The results of carried out experiments are presented as well.

Keywords: stereo image, epipolar geometry, corresponding points, disparity.

УДК 330.43

С. В. Вохмянин

ИСПЫТАНИЕ АЛГОРИТМА МЕТОДА «ГУСЕНИЦА-SSA» ДЛЯ ВОССТАНОВЛЕНИЯ ВРЕМЕННОГО РЯДА

Рассмотрен базовый алгоритм метода «Гусеница-SSA» и проведены его испытания.

Ключевые слова: выделение тренда, нахождение периодик, устранение шума, разложение временного ряда на компоненты.

Одной из важнейших задач в анализе временных рядов является отделение тренда и периода от шума. Данная статья посвящена исследованию мощного и быстроразвивающегося метода анализа временных рядов «Гусеница-SSA» [1].

Рассмотрим временной ряд F:

$$f_0, f_1, ..., f_{N-1}, \quad (1)$$

где N – его длина. В дальнейшем будем предполагать, что ряд F – ненулевой.

Алгоритм метода «Гусеница-SSA» состоит из четырех последовательно выполняемых шагов: вложения, сингулярного разложения, группировки и диагонального усреднения.

На *первом шаге* процедура вложения переводит исходный временной ряд F в последовательность многомерных векторов, которая называется *траекторной матрицей*.

Для анализа временного ряда выбирается целочисленный параметр L, именуемый *длиной окна*, такой что $1 < L < N$. При этом образуется $K = N - L - 1$ векторов вложения:

$$X_i = (f_{i-1}, f_i, ..., f_{i+L-2})^T, \ 1 \le i \le K .$$

Эти векторы образуют траекторную матрицу ряда F, столбцами которой являются скользящие отрезки ряда длины L: с первой точки по L-ю, со второй по $(L + 1)$-ю и т. д.:

$$X = [X_1 : X_2 : ... : X_K] = \begin{pmatrix} f_0 & f_1 & ... & f_{K-1} \\ f_1 & f_2 & ... & f_K \\ ... & ... & ... & ... \\ f_{L-1} & f_L & ... & f_{N-1} \end{pmatrix}. \quad (2)$$

Существует взаимно однозначное соответствие между матрицами размерности $L \times K$ вида (2) и рядами (1) длины $N = L + K - 1$ [1].

Результатом *второго шага* является сингулярное разложение траекторной матрицы (2) в сумму элементарных матриц.

Пусть $S = X \cdot X^T$. Обозначим через $\lambda_1, \lambda_2, \ldots, \lambda_L$ собственные числа матрицы S, взятые в неубывающем порядке, а через $U_1, U_2, \ldots, U_L$ ортонормированную систему собственных векторов матрицы S, соответствующих упорядоченным собственным числам. Тогда сингулярное разложение траекторной матрицы X может быть записано следующим образом:

$$X = \sum V_i , \quad (3)$$

где $V_i = U_i \cdot U_i^T \cdot X$, $I = 1, \ldots, L$. Учитывая, что каждая из матриц V_i имеет ранг 1, назовем их элементарными матрицами [1].

Предположим, что исходный временной ряд является суммой нескольких рядов, что позволяет при некоторых условиях определить по виду собственных чисел и собственных векторов, какие это слагаемые и какой набор элементарных матриц соответствует каждому из них.

На *третьем шаге* на основе разложения (3) множество индексов $\{1, 2, \ldots, L\}$ делится на m непересекающихся подмножеств $I_1, I_2, \ldots, I_m$. Тем самым разложение (3) может быть записано в виде

$$X = \sum_{i=1}^{m} Y_i , \quad (4)$$

где $Y_i = \sum_{k \in I_i} V_k$ – результирующие матрицы для каждого подмножества I_i, $I = 1, \ldots, m$.

Фактически именно на этом шаге происходит разделение исходного ряда (1) на шумы, тренд и периодики. Основным критерием группировки является значимость