ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2024, том 13, № 2

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2024, volume 13, no. 2

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2024
Т. 13, № 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2024
Vol. 13, no. 2

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

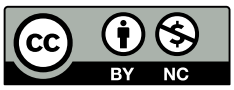
South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents



УДК 519.6 **DOI: 10.14529/cmse240201**

РЕКОНСТРУКЦИЯ ИЗОБРАЖЕНИЯ ПО МЕТОДУ ОБРАТНОГО ПРОЕЦИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ВЕЙВЛЕТ-ФИЛЬТРАЦИИ ПРОЕКЦИОННЫХ ДАННЫХ В РЕНТГЕНОВСКОЙ КОМПЬЮТЕРНОЙ ТОМОГРАФИИ

© 2024 Е.Н. Симонов, К.М. Виноградов

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: e.n.simonov@yandex.ru, vinogradovkm@susu.ru

Поступила в редакцию: 14.12.2022

В статье представлен метод уменьшения ошибки реконструкции изображения для рентгеновской компьютерной томографии путем применения вейвлет-фильтрации зашумленных проекционных данных. Вейвлет-преобразование и основанное на нем вейвлет-фильтрация одномерных сигналов дает возможность определять конкретное место соответствия частотной и временной (в данном случае пространственной по координате детекторов) области. Это позволяет однозначно определять переход из частотной области в пространственную и обратно. Для фильтрации проекционных данных используется вейвлет-преобразование, которое дает возможность через коэффициенты, определяющие масштабирующие функции и функции вейвлетов определять в частотной и пространственной области место шума в зашумленном сигнале и осуществлять выделение не зашумленного сигнала путем назначения порогов фильтрации на вышеуказанные коэффициенты. Для усиления фильтрующих свойств вейвлет-преобразования предложено разбивать проекционные данные на интервалы, для каждого из которых определяются свои коэффициенты. Вейвлет-фильтрация проводится с использованием вейвлетов Добеши. Результаты исследований были подтверждены математическим моделированием зашумленных проекционных данных, их вейвлет-фильтрации и реконструкции по ним тестового томографического изображения. Математическая модель тестового объекта исследования и разработанный авторами программный реконструктор томографического изображения позволили осуществлять моделирование прямой (получение проекционных данных по тестовому объекту), обратной (получение тестового томографического изображения по проекционным данным объекта) задач томографии и осуществлять сравнительный анализ качества реконструкции изображения с «идеальными» и зашумленными проекционными данными.

Ключевые слова: рентгеновская компьютерная томография, проекционные данные, вейвлеты.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Симонов Е.Н., Виноградов К.М. Реконструкция изображения по методу обратного проецирования с использованием вейвлет-фильтрации проекционных данных в рентгеновской компьютерной томографии // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 5–22. DOI: 10.14529/cmse240201.

Введение

Повышение точности реконструкции изображений в рентгеновской компьютерной томографии, со времени ее открытия и получения Нобелевской премии, является одной из важных задач в достижении пространственного и плотностного разрешения томографических систем. Точность реконструкции зависит от факторов:

1. Устойчивости решения томографического уравнения через обратное преобразование Радона (обратная задача томографии), т.к. эта задача относится к классу некорректно поставленных задач и требует регуляризации в решении.
2. Погрешности рентгено-оптического тракта томографа при измерении сигналов детекторной системой, вызванной различными причинами: погрешностью самой электрон-

ной системы измерения, квантовыми шумами детектора, неоднородностью поля излучения источника, изменением геометрии тракта при сканировании объекта.

В данной статье рассматривается задача уменьшения влияния погрешности второго фактора, который непосредственно определяет точность формирования проекционных данных, участвующих в реконструкции томографического изображения. Влияние первого фактора в статье не рассматривается. Считается, что алгоритм реконструкции является корректным и регуляризированным.

Структура данной работы такова. Раздел 1 содержит обзор работ, посвященных фильтрации томографических проекционных данных. В разделе 2 дано описание разработанного программного реконструктора, как инструмента исследования влияния шумов на качество томографического изображения. Раздел 3 содержит предлагаемый метод фильтрации проекционных данных. Вычислительные эксперименты и их результаты представлены в разделе 4. Последний раздел содержит заключение и выводы исследования.

1. Обзор работ

Вопросам фильтрации в рентгеновской компьютерной томографии посвящено достаточно много работ. Однако основная их часть посвящена вопросам фильтрации самого алгоритма реконструкции, чтобы он был устойчивым к малым флюктуациям проекционных данных, т.е. фильтрация используется для регуляризации обратного преобразования Радона при решении основного уравнения рентгеновской томографии [1–4, 22], или вопросам фильтрации уже полученного томографического изображения [5–12].

В работах [9–16] было показано, что погрешности детекторной системы измерения проекционных данных такие, как квантовый шум детекторов, статистические скачки измерения единичный детекторов или группы детекторов (единичных проекционных данных или группы), статистические пространственные колебания поля источника излучения или колебания геометрических параметров рентгенооптического тракта (горизонтальное или вертикальное смещение детекторов) приводят к различным шумам и артефактам на томографическом изображении. Уменьшить их влияние на качество изображения с применением пространственных и частотных методов фильтрации изображений практически не представляется возможным [6–9]. Поэтому с таким влиянием указанных погрешностей на качество изображения необходимо бороться на самом начальном этапе томографического процесса, на этапе формирования проекционных данных.

В работах [10, 11] предлагалось осуществлять фильтрацию проекционных данных с применением преобразования Фурье (фурье-фильтрация). Однако результаты исследования неудовлетворительные. Причинами неточной фильтрации проекционных данных могут быть:

1. преобразование Фурье, базисной функцией которого являются гармонические колебания, может давать значительные погрешности при скачкообразных измерительных данных с детекторов (эффект Гиббса);
2. Фурье преобразование имеет и другой фундаментальный недостаток, как невозможность указать соответствие локальной частотной области сигнала его локальной временной (пространственной) области. Это отражается на диагностике (определении) локальных областей зашумленных детекторов (измерений), определении частотного спектра их фильтрации.

Работ по фильтрации проекционных данных в компьютерной томографии достаточно мало [9, 11–16]. Они имеют специальный характер, направлены на уменьшение влияния конкретного фактора, приводящего к специфическому артефакту на изображении, как правило, физическими методами.

Главной причиной проблемы фильтрации проекционных данных в томографии является сложный многомерный алгоритм реконструкции томографического изображения [9]. Проследить путь от шумящего по какой-либо причине конкретного детектора (группы детекторов) до какого-то конкретного пикселя (группы пикселей) томографического изображения не представляется возможным в силу многократного преобразования проекционных данных (сотни тысяч операций) с различными их весами в силу физики поглощения излучения, их усиления, сверточного преобразования (усечения высоких частот). На погрешность конкретного пикселя изображения влияют все проекции (шумящие и не шумящие), полученные по всем детекторам и ракурсам, и определение влияния шума проекционных данных (или какой-то их части) на точность реконструкции изображения является сложной задачей. Это является главным отличием от «классического» измерительного канала с помехой [10].

Общего математического подхода к фильтрации проекционных данных, который бы давал возможность выделять требуемую структуру изменения сигнала, определять локальные области зашумленных детекторов и применять к ним локальные фильтры, в настоящее время не выработано. Такими возможностями, в определенной мере, обладает фильтрация на основе вейвлет-преобразования [17]. Фильтрация одномерных сигналов на основе вейвлет-преобразования пока не нашла должного применения в практических приложениях, как фурье-фильтрация. Основная причина, видимо, достаточно сложный математический аппарат вейвлет-преобразования и практическое отсутствие разработанных прикладных методик использования его уникальных свойств.

2. Описание инструмента исследования — программного реконструктора томографического изображения

Для исследования влияния погрешности второго фактора на томографическое изображение авторами статьи был разработан программный реконструктор томографиче-ских рентгеновских изображений, позволяющий решать прямую задачу томографии, т.е. определять массив проекционных данных для различных ракурсов облучения и отсчетов линейки детекторов по модели объекта исследования, и обратную задачу, т.е. по заданному массиву проекционных данных проводить томографическую реконструкцию изоб-ражения модели объекта.

Таким образом, программный реконструктор, как инструмент исследования, дает возможность по модели объекта исследования получать «идеальные» проекционные данные, накладывать на них общие и локальные (по ряду ракурсов или отсчетов) шумы, проводить реконструкцию томографического изображения на зашумленных проекционных данных, оценивать качество изображения (ошибку реконструкции) относительно «идеального» изображения.

При разработке программного реконструктора использовались теоретические основы рентгеновской компьютерной томографии [1, 9]. С помощью рентгеновской трубки (источника) получают пучки рентгеновских лучей интенсивности I_0, которые проходят через объ-

ект исследования и регистрируются детекторами. На рис. 1 показана параллельная схема сканирования рентгеновского пучка излучения.

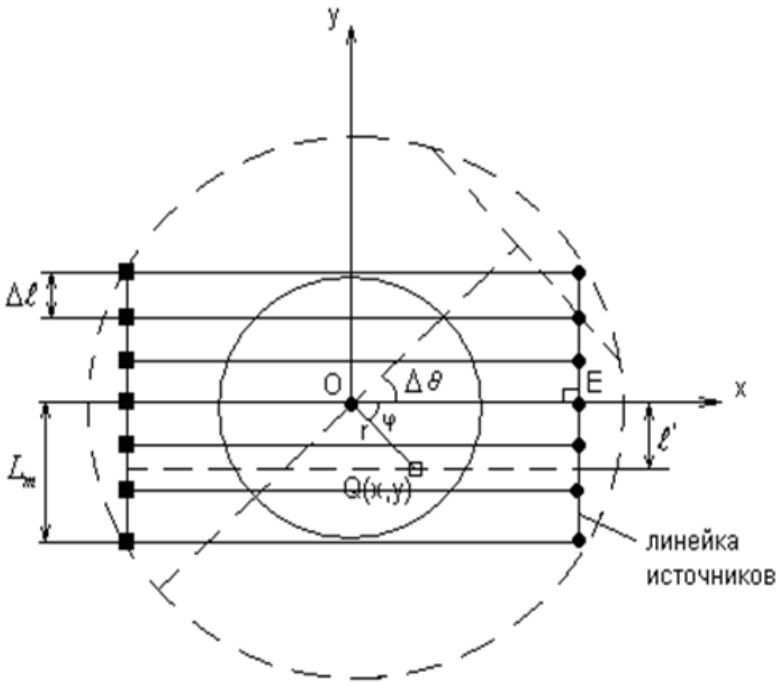


Рис. 1. Параллельная схема сканирования рентгеновского пучка излучения

Закон Бугера—Ламберта—Бера определяет ослабление монохроматического пучка излучения при распространении его в объекте исследования:

$$I(l) = I_0 \exp(-\int_{-\infty}^{+\infty} \mu_l dl), \tag{1}$$

где I_0 — интенсивность входящего пучка в объект исследования, $I(l)$ — интенсивность выходного пучка из объекта исследования, dl — элементарное расстояние, которое проходит рентгеновский луч через вещество по прямой l, μ_l — линейный коэффициент ослабления рентгеновского излучения на элементарном расстоянии dl.

Обозначив через $q(l,\theta)$ проекционные данные для положения l детектора и угла поворота θ относительно начального положения, получим из (1):

$$q(l,\theta) = -\ln(\frac{I(l,\theta)}{I_0}) = (\int_{-\infty}^{+\infty} \mu_l dl)_{l,\theta}. \tag{2}$$

Запись (2) есть основное уравнение рентгеновской компьютерной томографии и означает, что интеграл определяется для каждого значения l и θ.

В полученных проекционных данных присутствует погрешность, потому что определенный уровень погрешности всегда имеет место в любом детекторной электронной системе, которая измеряет интенсивность рентгеновского пучка на входе I_0 и выходе $I(l) = I(l,\theta)$ объекта исследования. Учитывая, что множество вышеуказанных факторов, влияющих на уровень погрешности, являются независимыми и ни один из них не преобладает над остальными, распределение погрешности целесообразно выбрать нормальным, то есть в виде гауссовского шума.

Согласно работе [9], зашумленные проекционные данные можно представить в виде суммы проекционных данных без шума и аддитивного гауссовского шума

$$q' = q + \eta, \tag{3}$$

где q' — зашумленные проекционные данные; q — проекционные данные без шума; η — гауссовский шум. Реконструкция томографического изображения в параллельных лучах для интегрального аналитического алгоритма обратного проецирования с фильтрацией сверткой (Filtered Back Projection FBP) [1] представляет собой определение в (2) μ_l в координатах (x, y). Такое решение представляется как обратное преобразование Радона уравнения (2)

$$\mu(x, y) = \int_0^x \int_{-\infty}^{+\infty} g(l' - l) q'(l, \theta) dl d\theta, \tag{4}$$

где $g(l' - l)$ — ядро свертки (сворачивающая функция) для параллельной геометрии, где $q(l) = \frac{1}{4\pi} \int_{-\infty}^{+\infty} |w| W(w) e^{iws} dw$, где w — пространственная частота, $W(w)$ — окно фильтрации, $q'(l, \theta)$ — исходные зашумленные проекционные данные, l' — расстояние между точкой $Q(x, y)$ и прямой OE (рис. 1). В работе [9] автором данной статьи для множества различных окон фильтрации $W(w)$ (прямоугольного окна, синусного, параболического и др.) определены сворачивающие функции $g(l)$ (ядра свертки). Так, например, для прямоугольного окна сворачивающая функция будет иметь вид:

$$g(l) = \frac{1}{2\pi^2} \Big(\frac{w_{np}}{l} \sin(w_{np} l - \frac{2}{l^2} \sin^2(\frac{w_{np} l}{2}) \Big), \tag{5}$$

где пространственная частота $w_{np} = \frac{1}{2\Delta l}$. Этот вид сворачивающей функции используется в программном реконструкторе для получения томографического изображения. Схема получения томографического изображения функции (4) (обратная задача томографии) по заданным проекционным данным (2) показана на рис. 2.

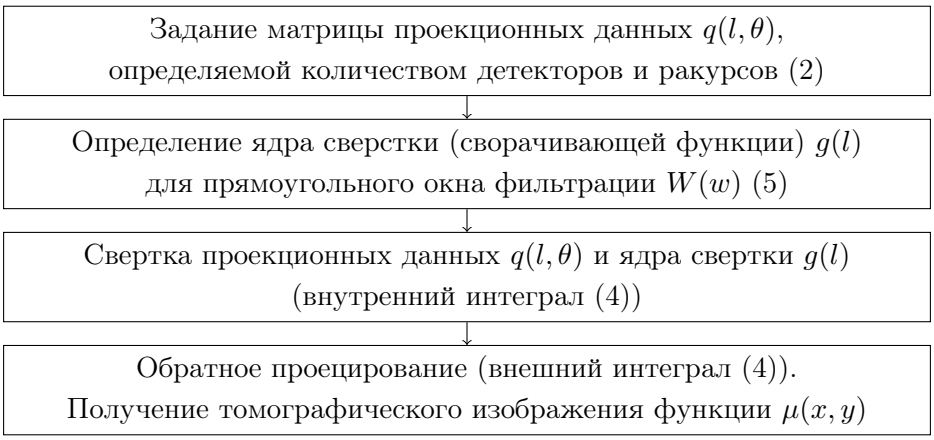


Рис. 2. Схема получения томографического изображения

Программный код реконструкции томографического изображения по заданным проекционным данным на языке C++ отражен в [21]. Схема получения проекционных данных (2) (прямая задача томографии) показана на рис. 3.

Программный код моделирования проекционных данных для параллельной геометрии сканирования на языке C++ отражен в [21].

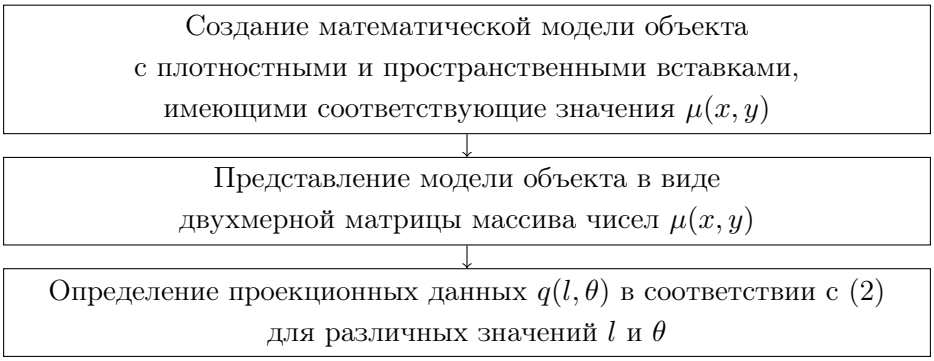


Рис. 3. Схема получения проекционных данных

3. Предлагаемый метод фильтрации

Для фильтрации зашумленных проекционных данных использовалось их вейвлет-преобразование. Вейвлеты — это обобщенное название семейств математических функций определенной формы, которые локальны во времени (в пространстве) по амплитуде и частоте и в которых все функции получаются из одной базовой (порождающей) функции посредством ее сдвигов и растяжений по оси времени (пространства) [17]. Дискретное вейвлет-преобразование любой функции $f(x)$ можно представить, как

$$\begin{aligned} P_j f(x) &= \sum_n a_{j,n}\phi_{j,n}(x) = \sum_n a_{j+1,n}(x) + \sum_n d_{j+1,n}\Psi_{j+1,n}(x) = \\ &= \sum_n a_{j+2,n}\Psi_{j+2,n}(x) + \sum_n d_{j+2,n}\Psi_{j+2,n}(x) + \sum_n d_{j+1,n}\Psi_{j+1,n}, \end{aligned} \tag{6}$$

причем $\lim\limits_{j\to-\infty} P_j f(x) = f(x)$.

Такое преобразование можно продолжить далее (рис. 4).

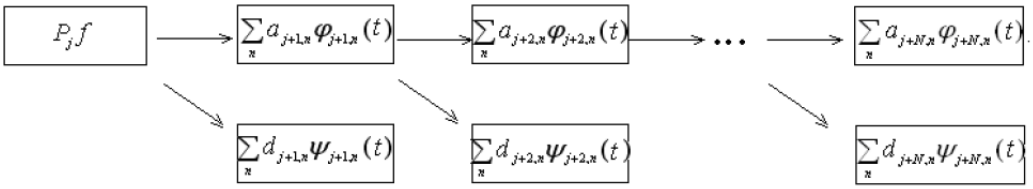


Рис. 4. Схема вейвлет-преобразования

Функция $\varphi(x)$ называется масштабирующей функцией и определяется как

$$\varphi(x) = \sum_n h_n\varphi_{-1,n}(x) = \sqrt{2}\sum_n h_n\varphi(2x-n), \tag{7}$$

где h_n — коэффициенты фильтра для применяемого вейвлета, $n = \{0, 1, 2, \dots\}$.

Функция $\varphi(x)$ называется функцией вейвлета и определяется как:

$$\psi(x) = \sqrt{2}\sum_n g_n\varphi(2x-n), \tag{8}$$

где $g_n = (-1)^2 h_{1-n}$.

В (6) $\{a_j\}$, $\{d_j\}$ — коэффициенты дискретного вейвлет-преобразования, где j — уровень разложения сигнала на низкочастотные и высокочастотные составляющие. Соответственно,

$$\phi_{j,n} = 2^{-j/2}\phi(2^{-j}x - n) = \sum_n h_n \phi_{j-1,1+2n}(x), \tag{9}$$

$$\psi_{j,n} = 2^{-j/2}\psi(2^{-j}x - n) = \sum_n g_n \psi_{j-1,1+2n}(x). \tag{10}$$

Дискретное вейвлет-преобразование проводит перевод последовательности $\{a_{j,n}\}$ в последовательности $\{a_{j+1,n}, d_{j+1,n}\}$. Множества $\{h_n\}$, $\{g_n\}$ позволяют по формулам (8)–(10) численно находить коэффициенты дискретного вейвлет-преобразования. Применяя преобразование (6) для нашей задачи, получим, что проекционные данные $q'(l,\theta)$ можно представить как

$$q'(l,\theta) = \sum_{n=1}^{\infty} a_{j,n}\phi_{j,n}(l,\theta) + \sum_{j=1}^{\infty}\sum_{n=1}^{\infty} d_{j,n}\Psi_{j,n}(l,\theta), \tag{11}$$

где $a_{j,n}$, $d_{j,n}$ — соответственно, аппроксимирующие и детализирующие коэффициенты вейвлет-преобразования; $\phi_{j,n}$ — масштабирующая функция; $\psi_{j,n}$ — функция вейвлета. Таким образом, анализируя (6) и рис. 3, аппроксимирующую часть функции можно многократно разложить на еще более сглаженную (аппроксимирующую) часть и детализирующую часть.

Считаем, что шум содержится в высокочастотных компонентах, т.е. находится в коэффициентах вейвлет-преобразования, отвечающих за малые масштабы (d). Фильтрация сводится к обрезанию «высоких частот» всего массива проекционных данных или его локальной части, т.е. приравниванию к нулю коэффициентов d ниже заданного значения (порога). В результате ожидается, что структура обрабатываемых данных, которая лежит в основе проекционного сигнала, сохранится и проявится после очищения от общего или локального шума [18–20].

Учитывая фундаментальное свойство вейвлет-преобразования, как соответствие временной (пространственной) и частотной областей сигнала в каждой точке, имеется возможность локального анализа сигнала и его шума. Чтобы усилить локальные свойства вейвлет-преобразования, авторами работы было предложено разбиение проекционного сигнала на интервалы с высоким изменением амплитуды, где амплитуда шума мало изменяет общую тенденцию сигнала, и фильтрация шума возможна без искажения сигнала, и интервалы с малым изменением амплитуды сигнала, где амплитуда шума изменяет общую тенденцию сигнала, и фильтрация шума может быть не возможна без искажения сигнала. Для каждого интервала проводится вейвлет-преобразование и определяются свои аппроксимирующие и детализирующие d коэффициенты.

Критерием разбиения проекционного сигнала на интервалы является отношение

$$K = \frac{{A_N}^{max}}{\Delta A}, \tag{12}$$

где ${A_N}^{max}$ — максимальная амплитуда шума в интервале, ΔA — изменение амплитуды проекционного сигнала без шума в интервале. Для интервала, где коэффициент $0 < K \leqslant 1$, фильтрация шума возможна без искажения сигнала более жесткими фильтрами и порогами, дающие меньший эффект сглаживания сигнала. Для интервала, где $K > 1$, фильтрация шума требует применение мягких фильтров и порогов.

Таким образом, алгоритм очистка проекционных данных с применением вейвлет-преобразования сводится к следующим шагам.

1. Проведение вейвлет-преобразования для каждого интервала проекционных данных.
2. Задание порога для коэффициентов вейвлет-преобразования d в соответствии с их уровнем разложения j для всего сигнала или его интервала.
3. Проведение фильтрации для интервалов проекционных данных с заданным порогом.
4. Восстановление проекционных данных по измененным коэффициентам d.

Существуют различные виды задания порога для коэффициентов вейвлет-преобразования d [18]. Пусть d_j, d'_j — вейвлет-коэффициенты некоторого уровня j до фильтрации и после, соответственно, t — значение порога, тогда
мягкая вейвлет-фильтрация определяется как

$$d'_j = \operatorname{sign}(d_j)\max(0, |d_j| - t), \tag{13}$$

жесткая вейвлет-фильтрация определяется как

$$d'_j = \begin{cases} 0, |d_j| \leqslant t \\ d_j, |d_j| > t \end{cases}, \tag{14}$$

аффинная вейвлет-фильтрация имеет следующее определение:

$$d'_j = \begin{cases} 0, |d_j| < t/2 \\ 2d_j + t, -t \leqslant d_j \leqslant -t/2 \\ 2d_j - t, t \leqslant d_j \leqslant t \\ d_j, |d_j| > t \end{cases}. \tag{15}$$

Значение порога t играет важную роль. Слишком малое значение порога не может удалить шумовые составляющие, слишком большое значение может удалить полезные составляющие проекционных данных. Существуют следующие способы поиска порога [18–20]:

1. Универсальный порог
$$t = \sigma\sqrt{2\log_2(n)}, \tag{16}$$
где n — число отсчетов сигнала, σ — уровень шума.
2. Минимаксный порог
$$t = \sigma t_n, \tag{17}$$
где t_n выбирается, исходя из минимаксного правила, т.е. $\inf(\sup(E(f, f'))$, где $E(f, f')$ — оценка среднеквадратичной ошибки между искомой и найденными функциями.
3. Порог, устанавливаемый на основании правила *Birge—Massart*,
$$t = |c_k|, \tag{18}$$
где $k = \min(-\sum {c_j}^2 + 2\sigma^2 n'(\alpha + \log_2(n/n'))$, c_j — вейвлет-коэффициенты d_j, отсортированные в порядке убывания своих абсолютных значений, n' — число отсчетов сигнала, $1 \leqslant n' \leqslant n$.
4. Порог, устанавливаемый на основании правила *Donoho—Johnstone*
$$t = \frac{\sigma}{\sigma_x}, \tag{19}$$

где σ_x — локальная ошибка исследуемой функции, σ определяется как

$$\sigma = \frac{med(|W_1(f')|)}{0.6745}, \tag{20}$$

где $W_1(f')$ — нижний уровень детализирующих вейвлет-коэффициентов.

Реконструируя функцию $\mu(x, y)$ по формуле (4) по зашумленным проекциям $q'(l, \theta)$ и фильтрованным $q(l, \theta)$, можно оценить качество томографического изображения и сделать сравнительный анализ.

Таким образом, последовательность реконструкции томографического изображения с использованием вейвлет-фильтрации проекционных данных будет следующей (рис. 5).

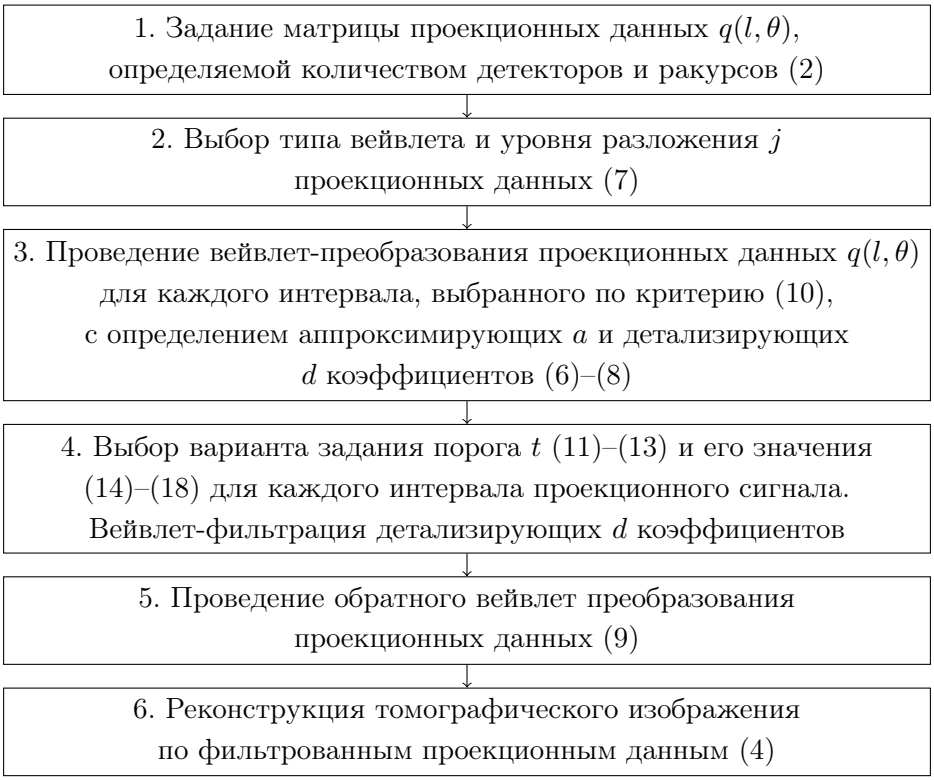


Рис. 5. Схема реконструкции томографического изображения с использованием вейвлет-фильтрации проекционных данных

Поясним принципиальные особенности вейвлет-фильтрации проекционных данных (шаг 4 схемы рис. 5). Проекционные данные $q(l, \theta)$ в параллельных рентгеновских лучах (рис. 1) представляются в виде матрицы $N \times M$ (рис. 6), где N — количество детекторов, $l = \delta l \times n$, n — порядковый индекс (номер) детектора, $n = \{1, 2, 3, \ldots, N\}$, M — количество ракурсов облучения, $\theta = \delta\theta \times m$, m — порядковый индекс (номер) ракурса, $m = \{1, 2, 3, \ldots, M\}$.

Из матрицы проекционных данных объекта исследования (рис. 6) для каждого ракурса $m = \{1, 2, 3, \ldots, M\}$ выделяют одномерный проекционный сигнал $q(l)$ и определяют интервалы для вейвлет-анализа в соответствии с критерием (10).

На рис. 7 показан типовой одномерный проекционный сигнал $q(l)$ с шумом и интервалы разбиения сигнала в соответствии с критерием (10). В интервалах 1–7, 9 значения $K < 1$, в интервале 8 значения $K > 1$. Это значит, что для интервалов 1–7, 9 целесообразно применять жесткую или аффинную вейвлет-фильтрацию (9), (10), т.к. шум мало оказывает

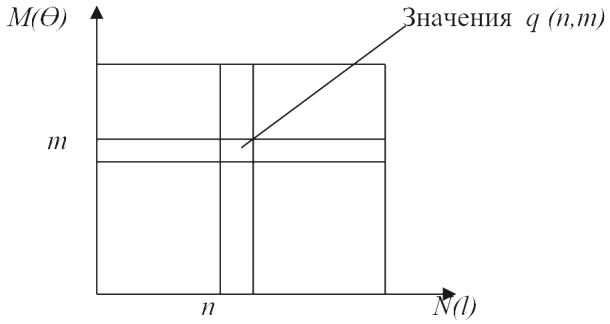


Рис. 6. Матрица проекционных данных

влияние на общую тенденцию изменения сигнала. Порог для этих интервалов возможен по (14), (15), т.к. здесь не требуется адаптации к изменению сигнала, ибо это изменение явно выражено. Для интервала 8 целесообразно применять мягкую вейвлет-фильтрацию (11), а порог возможен по правилу *Birge—Massart* (16) или правилу *Donoho—Johnstone* (17), т.к. сигнал сильно зашумлен, изменения его малозаметны на фоне шума, и здесь требуются некоторые элементы адаптации к изменению сигнала при выборе типа вейвлет-фильтрации и порога, что и заложены в (11), (16) и (17). В противном случае, жесткая фильтрация и «жесткие» пороги могут удалить полезные составляющие сигнала.

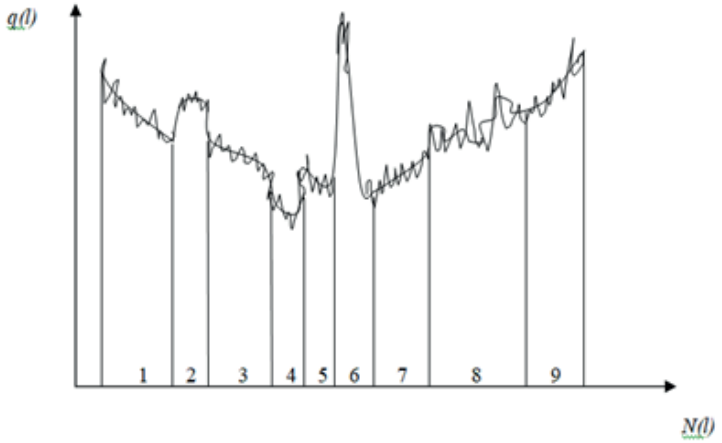


Рис. 7. Типовой одномерный проекционный сигнал $q(l)$ с шумом

4. Вычислительные эксперименты

Для проведения вычислительного эксперимента использовался разработанный авторами статьи программный реконструктор томографического изображения. Реконструктор позволяет получать матрицы изображений размером 512×512 пикселей. Объем проекционных данных может моделироваться для $M = 600$ ракурсов облучения (угол θ), количества единичных отсчетов (δl) в линейке детекторов $N = 512$, т.е. получать матрицу проекционных данных $M \times N$ (512×600).

Для количественной оценки шума на томографическом изображении использовалась среднеквадратическая ошибка:

$$E = \Big(\frac{\sum_i \epsilon \sum_j \epsilon (A_{i,j} - B_{i,j})^2}{\sum_i \epsilon \sum_j \epsilon (A_{i,j}} \Big)^{1/2}, \tag{21}$$

где A, B — значение матрицы исходного и реконструируемого изображений, i, j — номер соответствующего элемента (пикселя) в матрице изображения, ϵ — количество элементов (пикселей) в матрице.

Анализ влияния вейвлет-фильтрации проекционных данных на томографическое изображение был проведен для следующих вариантов:

1. вейвлет-фильтрация без разбиения одномерного проекционного сигнала на интервалы с заданием мягкой (Мвф) и жесткой (Жвф) фильтрации, с заданием порогов по правилам Birge—Massart (BM) и Donoho—Johnstone (DJ);
2. вейвлет-фильтрация с разбиением одномерного проекционного сигнала на интервалы в соответствии с критерием (10) с заданием типов фильтрации и порогов, что в п. 1.

Для фильтрации в обоих вариантах использовался вейвлет Добеши типа 8 [17], уровень разложения проекционных данных взят $j = 3$.

Для анализа была разработана тестовая математическая модель объекта с высоко контрастными прямоугольными и низко контрастными круговыми вставками (рис. 8). На модели объекта проверялась точность работы программного реконструктора. Из рис. 8 видно, что точность реконструкции высокая: отсутствуют на томографическом изображении шумы и артефакты.

а) изображение модели объекта исследования

б) реконструированное томографическое изображение модели

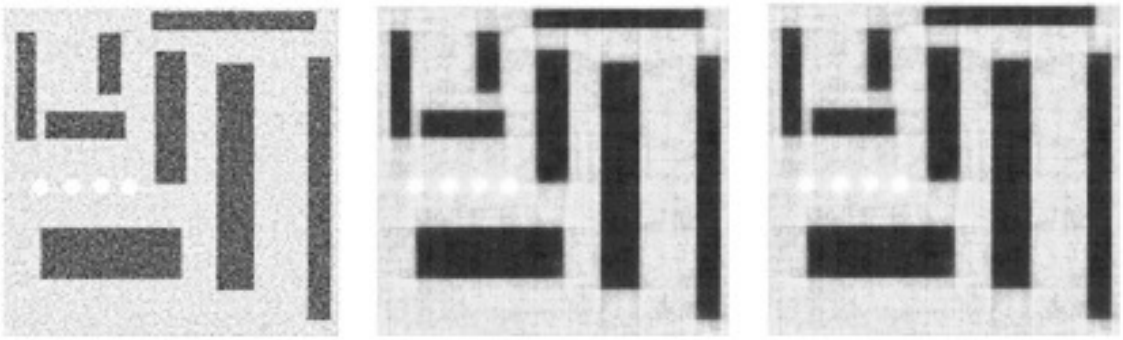
Рис. 8. Изображение модели объекта исследования и реконструированное томографическое изображение модели

Последовательность операций по оценке влияния вейвлет-фильтрации зашумленных проекционных данных (одномерных проекционных сигналов $q(l)$ для каждого ракурса) на качество томографического изображения была следующей.

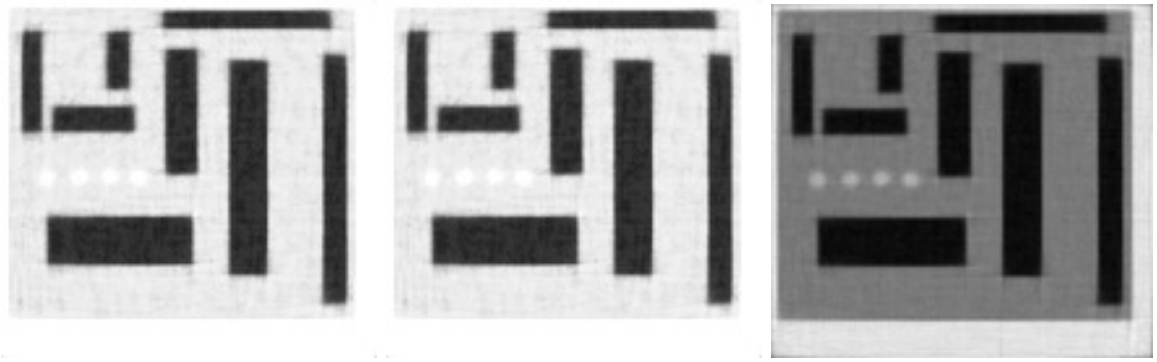
1. По исходному тестовому изображению (рис. 8) моделировались идеальные проекционные данные в соответствии с (2).
2. На идеальные проекции аддитивно накладывался гауссовский шум с получением зашумленных проекционных данных в соответствии с (3).
3. Проводилась реконструкция изображения с зашумленными данными в соответствии с (4).
4. Проекционные данные разбивались на интервалы в соответствии с критерием (10).
5. Для каждого интервала определялись детализирующие коэффициенты, тип их фильтрации в соответствии с (11)–(14) и пороги в соответствии с (14)–(18).

6. Проводилась реконструкция изображения в соответствии с (4) для отфильтрованных проекционных данных.
7. Проводился сравнительный анализ с оценкой (19) томографических изображений, реконструируемых без применения вейвлет-фильтрации (п. 3), с вейвлет-фильтрацией без разбиения проекционного сигнала на интервалы и с вейвлет-фильтрацией с разбиением проекционного сигнала на интервалы.

Результаты реконструкции и оценка качества томографического изображения — среднеквадратическая ошибка E, определенная по (19), приведены на рис. 9.



а) Без вейвлет-фильтрации ($E = 20.86\%$)
б) С использованием вейвлет-фильтрации без разбиения проекционного сигнала для мягкой фильтрации с заданием порогов по правилу Birge—Massart ($E = 7.42\%$)
в) С использованием вейвлет-фильтрации без разбиения проекционного сигнала для мягкой фильтрации по правилу Donoho—Johnstone ($E = 10.25\%$)



г) Для жесткой фильтрации с заданием порогов проекционного сигнала по правилу Birge—Massart ($E = 7.86\%$)
д) Для жесткой фильтрации с заданием порогов проекционного сигнала по правилу Donoho—Johnstone ($E = 10.01\%$)
е) С разбиением проекционного сигнала на интервалы по критерию (10) ($E = 6.43\%$)

Рис. 9. Реконструированные изображения

Результаты оценки качества изображений без применения, с применением вейвлет-фильтрации и с применением Фурье-фильтрации приведены в табл. 1.

При анализе исходного тестового томографического изображения и изображений, полученных при реконструкции с зашумленными проекционными данными, видно, что применение вейвлет-фильтрации проекционных данных может снизить ошибку реконструкции

Таблица 1. Результаты оценки качества изображений

<table>
<tr><td colspan="8">Погрешность реконструкции</td></tr>
<tr><td colspan="4">Вейвлет-фильтрация проекционных данных без разбиения проекционного сигнала на интервалы</td><td colspan="2">Вейвлет-фильтрация проекционных данных с разбиением проекционного сигнала на интервалы</td><td rowspan="3">Без фильтрации проекционного сигнала</td><td rowspan="3">Фурье-фильтрация проекционного сигнала</td></tr>
<tr><td colspan="4">Тип вейвлет-фильтрации</td><td colspan="2">Мвф</td></tr>
<tr><td colspan="2">Мвф</td><td colspan="2">Жвф</td><td rowspan="3">DJ</td><td rowspan="3">BM</td></tr>
<tr><td colspan="2">Тип порога</td><td colspan="2">Тип порога</td></tr>
<tr><td>DJ</td><td>BM</td><td>DJ</td><td>BM</td></tr>
<tr><td>7.42</td><td>7.86</td><td>10.25</td><td>10.01</td><td>6.43</td><td>6.48</td><td>20.86</td><td>20.4</td></tr>
</table>

(с 20.86 % до 6.43 %). Наилучшие результаты получены с вейвлет-фильтрацией проекционных данных с разбиением проекционного сигнала на интервалы. Фурье-фильтрация проекционных сигналов дает, в лучшем случае, качество томографического изображения на уровне без фильтрации [10, 11]. Это объясняется большой степенью сглаживания сигнала, а также фундаментальными недостатками Фурье преобразования. Методология практического применения вейвлет-фильтрации проекционных данных для конкретных рентгеновских томографических систем может быть следующей:

1. На тестовом калибровочном физическом фантоме плотностного разрешения [9] определяют:
   – идеальные не зашумленные проекционные данные по математической модели калибровочного физического фантома;
   – идеальные зашумленные проекционные данные калибровочного физического фантома на реальной томографической системе.
2. Зная изменения амплитуды не зашумленных проекционных данных (а), реальные зашумленные проекционные данные (б) разбивают на интервалы.
3. В каждом интервале проводят вейвлет-фильтрацию. С отфильтрованными проекционными данными проводят реконструкцию томографического изображения тестового калибровочного физического фантома на реконструкторе реальной томографической системы.
4. Проводят анализ реального томографического изображения:
   – если реальное томографическое изображение удовлетворяет требуемым характеристикам (разрешение, нелинейность, шум) [9], то параметры вейвлет-фильтрации проекционных данных, как тип фильтрации, тип порога, уровень разложения, задают в программное обеспечение обработки проекционных данных реальной рентгеновской томографической системы;
   – если реальное томографическое изображение не удовлетворяет требуемым характеристикам, то проводится изменение параметров вейвлет-фильтрации и пункты 3, 4 повторяют. Как правило, требуется 1–2 итерации.

Заключение

Повышение точности реконструкции изображений в рентгеновской компьютерной томографии является одной из важных задач в достижении пространственного и плотностного разрешения реальных томографических систем. Одним из главных факторов влияния на

достижение этих физических характеристик являются погрешности рентгено-оптического тракта томографа при измерении сигналов детекторной системой, вызванной различными причинами: погрешностью самой электронной системы измерения, квантовыми шумами детектора, неоднородностью поля излучения источника, изменением геометрии тракта при сканировании объекта. Все эти причины влияют на погрешность получения проекционных данных, по которым проводится реконструкция томографического изображения с применением сложного многомерного алгоритма.

На погрешность конкретного пикселя изображения влияют все проекции (шумящие и не шумящие), полученные по всем детекторам и ракурсам, и на этапе реконструкции томографического изображения фильтрацию шумящих детекторов практически осуществить невозможно. Существующие пространственные и частотные методы фильтрации изображений, в том числе и на основе Фурье преобразования, не дают желаемого результата при фильтрации томографических изображений и проекционных данных.

В работе представлен метод фильтрации сигналов на этапе получения проекционных данных по сигналам с детекторов с применением вейвлет-преобразования. Учитывая фундаментальные свойства вейвлет-преобразования, проекционный сигнал возможно представить в пространственной и частотной областях и определить соответствие этим областям для конкретной точки сигнала. Это соответствие определяется через апроксимирующие и детализирующие коэффициенты пространственной области, которые ответственны, соответственно, за низкие и высокие частоты сигнала в частотной области. Фильтруя детализирующие коэффициенты, ответственные за высокие частоты сигнала, посредством применения определенного вида фильтрации и типа порога, мы тем самым регулируем их влияние на исходный сигнал.

Исследования показали, что если проекционный сигнал разбить на интервалы с большими и малыми изменениями сигнала и провести отдельно в каждом интервале фильтрацию детализирующих коэффициентов, то адаптивные и фильтрующие свойства вейвлет-преобразования значительно усиливаются. В этом заключается суть представленного метода фильтрации проекционных сигналов. Представленный метод фильтрации проекционных данных с применением вейвлет-анализа показал, что ошибка реконструкции томографического изображения может быть снижена в 2–3 раза.

Метод фильтрации проекционных данных с применением вейвлет-анализа может найти практическое применение при проектировании томографических систем медицинского, промышленного и военного назначения. Требуются проведение дальнейших исследований по применения вейвлет-фильтрации проекционных данных, как двумерного массива сигналов, а также для уменьшения специфических артефактов на томографическом изображении в виде расплывчатых колец, полос, затенений.

Исследование выполнено за счет гранта Российского научного фонда № 23-21-10051 https://rscf.ru/project/23-21-10051/.

Литература

1. Луитт Р.М. Алгоритмы реконструкции с использованием интегральных преобразований // ТИИЭР. 1983. Т. 71, № 3. С. 125–148.

2. Луис А.К., Неттерер Ф. Математические проблемы реконструктивной вычислительной томографии // ТИИЭР. 1983. Т. 71, № 3. С. 111–125.

3. Barrett J.E., Keant N. Artifacts in CT: Recognition and Avoidance // Radio Graphics. 2004. Vol. 24. P. 1679–1691. DOI: 10.1148/rg.246045065

4. Арсенин В.Я., Криксин Ю.А., Тимонов А.А. Метод локальной регуляризации линейных операторных уравнений I рода и его приложения // Вычислительная математика и математическая физика. 1988. Т. 28, № 6. С. 793–808.

5. Пикалов В.В., Непомнящий А.В. Итерационный алгоритм с вэйвлет-фильтрацией в задаче двумерной томографии // Вычислительные методы и программирование. 2003. Т. 4, № 1. С. 244–253.

6. Воскобойников Ю.Е., Колкер А.Б. Комбинированные алгоритмы фильтрации зашумленных сигналов и изображений // Автометрия. 2002. № 4. С. 51–60.

7. Воскобойников Ю.Е., Бронников А.В. Адаптивный алгоритм фильтрации изображений и преобразование изображений в векторный формат // Автометрия. 1990. № 1. С. 124–132.

8. Воскобойников Ю.Е., Белявцев В.Г. Алгоритмы фильтрации изображений с адаптацией размеров апертуры // Автометрия. 1998. № 3. С. 81–89.

9. Симонов Е.Н. Физика визуализации изображений в рентгеновской компьютерной томографии. Челябинск: Изд-во ЮУрГУ, 2014. 479 с.

10. Ласьков В.В., Симонов Е.Н. Методы фильтрации изображений в рентгеновской компьютерной томографии // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2014. Т. 14, № 3. С. 29–33.

11. Laskov V.V., Simonov E.N. Reduction of ring artifacts in computer tomography // Biomedical Engineering. 2016. Vol. 49, no. 5. P. 274–277. DOI: 10.1007/s10527-016-9547-9.

12. Бессонов В.Б., Клонов В.В., Ларионов И.А., Староверов Н.Е. Разработка метода коррекции металлических артефактов при томографических исследованиях // Физические основы приборостроения. 2020. Т. 9, № 4(38). С. 54–59. DOI: 10.25210/jfop-2004-054059.

13. Бессонов В.Б., Потрахов Н.Н., Ободовский А.В. Рентгеновская томография // Фотоника. 2019. № 7. С. 688–693. DOI: 10.22184/1992-7296.FRos.2019.13.7.688.692.

14. Klonov V.V., Larionov I.A., Bessonov V.B., Baksheev I.K. Development of x-ray dose sensor // AIP Conference Proceedings. 2021. Vol. 2356, no. 1. P. 020013. DOI: 10.1063/5.0053146.

15. Staroverov N.E., Gryaznov A.Y., Bessonov V.B. Research of the possibility of using neural networks to identify areas of interest in tomographic data // AIP Conference Proceedings. 2020. Vol. 2250, no. 1. P. 020027. DOI: 10.1063/5.0013424.

16. Obodovskiy A.V., Bessonov V.B., Larionov I.A. Features of the practical application of microfocus x-ray tomograph in biomedical engineering // AIP Conference Proceedings. 2019. Vol. 2140, no. 1. P. 020049. DOI: 10.1063/1.5121974.

17. Daubechies I. The wavelet transform, time-frequency localization and signal analyses // IEEE Trans. Inform. Theory. 1990. Vol. 36, no. 5. P. 961–1005. DOI: 10.1109/18.57199.

18. Donoho D. Nonlinear solution of linear inverse problems by wavelet-vaguelette decompositions // Journal of Applied and Computational Harmonic Analysis. 1995. Vol. 2, no. 2. P. 101–126. DOI: 10.1006/acha.1995.1008.

19. Birge L., Massart P. From model selection to adaptive estimation // Festschrift for Lucien Le Cam / eds. by D. Pollard, E. Torgersen, G.L. Yang. Springer, 1997. P. 55–88. DOI: 10.1007/978-1-4612-1880-7_4.

20. Chang S., Yu B., Vetterli M. Spatially adaptive wavelet thresholding with context model-ing for image denoising // IEEE Transactions on Image Processing. 2000. Vol. 9, no. 9. P. 1522–1531. DOI: 10.1109/83.862630.

21. Симонов Е.Н. Реконструктор томографического изображения. Свидетельство о Государственной регистрации программы для ЭВМ № 2011612631. Зарегистрировано 31.04.2011 г.

22. Shi H., Luo S., Yang Z., Wu G. A Novel Iterative CT Reconstruction Approach Based on FBP Algorithm // PLOS One. 2015. Vol. 10, no. 9. P. e0138498. DOI: 10.1371/journal.pone.0138498.

Симонов Евгений Николаевич, д.т.н., профессор, кафедра техники, технологии и строительства, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Виноградов Константин Михайлович, к.т.н, доцент, кафедра техники, технологии и строительства, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240201

IMAGE RECONSTRUCTION BY THE METHOD OF REVERSE PROJECTION USING WAVELET FILTERING OF PROJECTION DATA IN X-RAY COMPUTED TOMOGRAPHY

© 2024 E.N. Simonov, K.M. Vinogradov

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: e.n.simonov@yandex.ru, vinogradovkm@susu.ru

Received: 14.12.2022

The article presents a method for reducing the error of image reconstruction for X-ray computer tomography by using wavelet filtering of noisy projection data. The wavelet transformation and the wavelet filtering of one-dimensional signals based on it makes it possible to determine a specific place of correspondence between the frequency and time (in this case, the spatial coordinate of the detectors) region. This makes it possible to uniquely determine the transition from the frequency domain to the spatial domain and vice versa. To filter the projection data, the wavelet transform is used, which makes it possible, through coefficients defining scaling functions and wavelet functions, to determine in the frequency and spatial domain the place of noise in a noisy signal and to isolate a non-noisy signal by assigning filtering thresholds to the above coefficients. To enhance the filtering properties of the wavelet transform, it is proposed to divide the projection data into intervals, for each of which its coefficients are determined. Wavelet filtering is carried out using Daubeshi wavelets. The research results were confirmed by mathematical modeling of noisy projection data, their wavelet filtering and reconstruction of the test tomographic image based on them. The mathematical model of the test object of the study and the software reconstructor of the tomographic image developed by the authors made it possible to simulate direct (obtaining projection data on the test object), reverse (obtaining a test tomographic image from the projection data of the object) tomography tasks and to carry out a comparative analysis of the quality of image reconstruction with "ideal" and noisy projection data.

Keywords: X-ray computed tomography, projection data, wavelets.

FOR CITATION

Simonov E.N., Vinogradov K.M. Image Reconstruction by the Method of Reverse Projection Using Wavelet Filtering of Projection Data in X-ray Computed Tomography. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 5–22. (in Russian) DOI: 10.14529/cmse240201.

References

1. Luitt R.M. Reconstruction algorithms using integral transformations. TIIER. 1983. Vol. 71, no. 3. P. 125–148.
2. Louis A.K., Netterer F. Mathematical problems of reconstructive computational tomography. TIIER. 1983. Vol. 71, no. 3. P. 111–125.
3. Barrett J.E., Keant N. Artifacts in CT: Recognition and Avoidance. Radio Graphics. 2004. Vol. 24. P. 1679–1691. DOI: 10.1148/rg.246045065.
4. Arsenin V.Ya., Kriksin Yu.A., Timonov A.A. Method of local regularization of linear operator equations of the first kind and its applications. Computational mathematics and mathematical physics. 1988. Vol. 28, no. 6, P. 793–808.
5. Pikalov V.V., Nepomnyashchy A.V. Iterative algorithm with wavelet filtering in a two-dimensional tomography problem. Computational methods and programming. 2003. Vol. 4, no. 1. P. 244–253.
6. Voskoboynikov Yu.E., Kolker A.B. Combined algorithms for filtering noisy signals and images. Autometry. 2002. No. 4. P. 51–60.
7. Voskoboinikov Yu.E., Bronnikov A.V. Adaptive image filtering algorithm and image conversion to vector format. Autometry. 1990. No. 1. P. 124–132.
8. Voskoboynikov Yu.E., Belyavtsev V.G. Image filtering algorithms with aperture size adaptation. Autometry. 1998. No. 3. P. 81–89.
9. Simonov E.N. Physics of image visualization in X-ray computed tomography. Chelyabinsk: SUSU Publishing House, 2014. 479 p.
10. Laskov V.V., Simonov E.N. Methods of image filtering in X-ray computed tomography. Bulletin of the South Ural State University. Computer Technologies, Control, Radio Electronics. 2014. Vol. 14, no. 3. P. 29–33.
11. Laskov V.V., Simonov E.N. Reduction of ring artifacts in computer tomography. Biomedical Engineering. 2016. Vol. 49, no. 5. P. 274–277. DOI: 10.1007/s10527-016-9547-9.
12. Bessonov V.B., Klonov V.V., Larionov I.A., Staroverov N.E. Development of a method for correcting metal artifacts in tomographic studies. Physical fundamentals of instrumentation. 2020. Vol. 9, no. 4(38). P. 54–59. DOI: 10.25210/jfop-2004-054059.
13. Bessonov V.B., Potrakhov N.N., Obodovsky A.V. X-ray tomography. Photonics. 2019. No. 7. P. 688–693. DOI: 10.22184/1992-7296.FRos.2019.13.7.688.692.
14. Klonov V.V., Larionov I.A., Bessonov V.B., Baksheev I.K. Development of x-ray dose sensor. AIP Conference Proceedings. 2021. Vol. 2356, no. 1. P. 020013. DOI: 10.1063/5.0053146.

15. Staroverov N.E., Gryaznov A.Y., Bessonov V.B. Research of the possibility of using neural networks to identify areas of interest in tomographic data. AIP Conference Proceedings. 2020. Vol. 2250, no. 1. P. 020027. DOI: 10.1063/5.0013424.

16. Obodovskiy A.V., Bessonov V.B., Larionov I.A. Features of the practical application of microfocus x-ray tomograph in biomedical engineering. AIP Conference Proceedings. 2019. Vol. 2140, no. 1. P. 020049. DOI: 10.1063/1.5121974.

17. Daubechies I. The wavelet transform, time-frequency localization and signal analyses. IEEE Trans. Inform. Theory. 1990. Vol. 36, no. 36. P. 961–1005. DOI: 10.1109/18.57199.

18. Donoho D. Nonlinear solution of linear inverse problems by wavelet-vaguelette decompositions. Journal of Applied and Computational Harmonic Analysis. 1995. Vol. 2, no. 2. P. 101–126. DOI: 10.1006/acha.1995.1008.

19. Birge L., Massart P. From model selection to adaptive estimation. Festschrift for Lucien Le Cam / eds. by D. Pollard, E. Torgersen, G.L. Yang. Springer, 1997. P. 55–88. DOI: 10.1007/978-1-4612-1880-7_4.

20. Chang S., Yu B., Vetterli M. Spatially adaptive wavelet thresholding with context modeling for image denoising. IEEE Transactions on Image Processing. 2000. Vol. 9, no. 9. P. 1522–1531. DOI: 10.1109/83.862630.

21. Simonov E.N. Certificate of state registration of software No. 2011612631 “Tomographic Image Reconstructor” issued on 31.04.2011.

22. Shi H., Luo S., Yang Z., Wu G. A Novel Iterative CT Reconstruction Approach Based on FBP Algorithm. PLOS One. 2015. Vol. 10, no. 9. P. e0138498. DOI: 10.1371/journal.pone.0138498.

УДК 004.93 **DOI: 10.14529/cmse240202**

СЕГМЕНТАЦИЯ 3D МОДЕЛЕЙ ДАННЫХ С ПОМОЩЬЮ МУЛЬТИМОДАЛЬНОГО ДИНАМИЧЕСКОГО ГРАФА CNN

© 2024 А.В. Вохминцев[1,2], В.Р. Аббазов[1], М.А. Романов[1]

[1] *Челябинский государственный университет*
(454001 Челябинск, ул. Братьев Кашириных, д. 129),
[2] *Югорский государственный университет*
(628012 Ханты-Мансийск, ул. Чехова, д. 16)
E-mail: vav@csu.ru, abbavar@yandex.ru, std.romanov.ma@gmail.com
Поступила в редакцию: 01.04.2024

В работе предложен метод семантической сегментации облаков точек в виде рельефа местности с использованием мультимодальной архитектуры сверточной нейронной сети на основе регулярного динамического взвешенного графа, которая позволяет получать точное решение задачи семантической сегментации, используя комбинацию геометрических и цветовых признаков точек. Метод может быть эффективно использован для разреженных, зашумленных, неоднородных и невыпуклых облаков точек. В работе было проведено компьютерное моделирование известных методов для семантической сегментации 3D данных с использованием эталонной коллекции данных ModelNet 40 и набора данных археологических памятников бронзового века Южного Зауралья, а именно данных, полученных в результате тахеометрической съемки комплекса археологических памятников в долине реки Синташта с использованием тахеометра Trimble 3300. Был проведен сравнительный анализ предложенного метода и современных методов 3D семантической сегментации с разными комбинациями входных признаков облаков точек, также в работе исследовано влияние на точность семантической сегментации способа формирования облака точек: в первом случае исследовалось облако точек из эталонного набора данных во втором случае применены варианты с использованием 3D регистрации на основе алгоритма ICP (iterative closest point).

Ключевые слова: сегментация 3D объектов, graph convolutional neural networks, регистрация облаков точек.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Вохминцев А.В., Аббазов В.Р., Романов М.А. Сегментация 3D моделей данных с помощью мультимодального динамического графа CNN // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 23–38. DOI: 10.14529/cmse240202.

Введение

При дешифрировании археологических памятников исследователи часто используют методы классификации и сегментации 3D данных на основе различных архитектур сверточных нейронных сетей. Методы можно поделить на непрямые, например, мультивидовые сверточный нейронные сети (MVCNN, multi-view convolutional neural network) [1] и прямые: PointNet [2], PointNet++ [3], на основе графа сверточной нейронной сети (GCNN, Graph Convolutional Neural Networks) [4]. На основе GCNN разработаны различные модификации: на основе динамического графа сверточной нейронной сети (DGCNN, dynamic graph CNN) [5], регулярного графа сверточной нейронной сети (RGCNN, regularized graph CNN) [6], ConvPoint [7]. Применение непрямых методов ограниченно, так как они обеспечивают хорошее качество семантической обработки только для простых полигональных моделей 3D данных, непрямые методы имеют низкую производительность и часто требовательны к памяти для хранения результатов. PointNet и PointNet++ основаны на локальной обработке точек, не используют геометрические отношения между точками и инвариантны

к перестановкам, данные методы находят широкое применение в промышленном дизайне. Модификации прямых методов на основе GCNN при семантической обработке 3D данных используют информацию о форме и поверхности объекта, процесс классификации и сегментации в них основан на выполнении множественных операций фильтрации и свертки сигналов на пространственных динамических графах в спектральной области. Например, в методе на основе RGGNN для описания связности компонент пространственного графа используется матрица Кирхгофа, построение которой имеет большую вычислительную сложность $O\left(n^3\right)$, что ограничивает применение метода при анализе крупномасштабных 3D сцен. GCNN и DGCNN плохо работают с локальными признаками объектов и находят основное применение при семантической классификации 3D объектов. Главный недостаток DGCNN и RGCNN связан с размерностью анализируемых облаков точек: методы сегментации на основе этих сетей хорошо подходят для семантической обработки эталонных коллекций 3D объектов, но плохо применимы к обработке реальных данных в виде плотных крупномасштабных облаков точек. Точность процедуры сегментации на основе различных модификаций GCNN зависит от формы объекта и способа формирования облака точек: хорошо сегментируются только исходные или идеально выровненные облака точек.

В последнее время был предложен ряд эффективных архитектур нейронных сетей для решения задачи классификации и семантической сегментации 3D данных, таких как, ConvPoint [7], KPConv [8], ShellNet [9] и Superpoint Transformer [10]. В табл. 1 представлены количественные результаты, которые иллюстрируют современное состояние методов для решения задачи 3D семантической сегментации с использованием коллекции данных Dales (A Large-scale Aerial LiDAR Data Set for Semantic Segmentation) [11]. В исследовании [7] предложен оригинальный метод, который использует непрерывные свертки при обработке облаков точек с неструктурированной природой и оказывается более эффективным, чем классические методы для 3D семантической сегментации [4–6]. ConvPoint в отличии от других методов способен работать с крупномасштабными наборами данных, которые могут включать тысячи плотных облаков точек большой размерности. Данная особенность и хорошие показатели качества по метрикам общей точности (OA, Overall Accuracy) и среднего пересечения над объединением (mIoU, mean intersection over union) делают данный метод кандидатом для решения поставленной в работе задачи по 3D семантической сегментации археологических памятников бронзового века. В исследовании [8] представлен новый подход к свертке точек KPConv для обработки облаков точек, который отличается деформируемой операцией свертки. Предложенный метод хорошо адаптируется к локальной геометрии облака точек и эффективен для обработки облаков точек различной плотности. KPConv был апробирован на многих наборах данных, кроме Dales, и для всех коллекций показывает хорошие показатели в терминах метрик OA и mIoU.

Методы PointNet и PointNet++ были одними из первых методов для 3D семантической сегментации, тем не менее они по-прежнему занимают высокие места в рейтинге качества, а для многих коллекций данных они показывают наилучшие результаты. В методе PointNet++ используется нейронная сеть с иерархической структурой для обработки наборов точек в метрическом пространстве. Данный метод использует метрику расстояния базового пространства и эффективно фиксирует локальные объекты в нескольких масштабах. PointNet++ показывает выдающиеся результаты для наборов данных с неоднородной плотностью выборки, а также отличается высокой производительностью.

В исследовании [9] предложена эффективная сверточная нейронная сеть ShellNet, использующая статистические данные из концентрических сферических оболочек для определения репрезентативных признаков в облаке точек при вычислении свертки. Предложенный подход позволяет с одной стороны существенно увеличить скорость обучения нейронной сети, а с другой стороны позволяет достигать самых современных результатов при классификации 3D объектов, сегментации частей объектов и семантической сегментации 3D сцен. Архитектура ShellNet подчеркивает потенциал оптимизированных нейронных сетей в обработке контекстуально сложных и крупномасштабных 3D сцен.

В исследовании Superpoint Transformer [10] представлен новый подход для эффективной семантической сегментации крупномасштабных 3D сцен на основе архитектуры трансформера. В архитектуре используется быстрый алгоритм для разбиения облаков точек на иерархические структуры суперпунктов, что позволяет значительно ускорить процедуру предварительной обработки 3D данных. В Superpoint Transformer используется разреженный механизм самовнимания для понимания взаимосвязей между суперпунктами в нескольких масштабах. Superpoint Transformer показывает высокое качество 3D семантической сегментации на тестовых наборах данных, обладает высокой производительностью, позволяет в компактной форме хранить модели в памяти.

Таблица 1. Результаты 3D семантической сегментации на коллекции данных Dales

Рейтинг	Модель	mIoU	Общая точность	Размер модели	Год
1	KPConv	81.1	97.8	14.1M	2019
2	Superpoint Transformer	79.6	97.5	212K	2023
3	SuperCluster	77.3	97.2	210M	2024
4	PointNet++	68.3	95.7	3.0M	2017
5	ConvPoint	67.4	97.2	4.7M	2018
6	SPG	60.6	95.5	280K	2018
7	PointCNN	58.4	97.2	N/A	2018
8	ShellNet	57.4	96.4	N/A	2019

Данная статья посвящена решению задачи повышения точности методов 3D сегментации облаков точек, в ней предложен метод сегментации на основе мультимодального динамического взвешенного графа DGCNN*, который использует лучшие идеи DGCNN и RGCNN, но свободен от ряда их ключевых недостатков. В этой статье мы исследуем методы сегментации, основанные на построении динамического графа CNN, методы на основе архитектуры трансформера не рассматриваются в данной статье.

В разделе 1 представлена архитектура мультимодального динамического граф DGCNN*, а также рассмотрены вопросы предварительной обработки облаков точек и их регистрации при построении 3D с разных углов обзора на сцене. В разделе 2 представлен алгоритм семантической сегментации 3D данных на основе DGCNN*. В разделе 3 представлены результаты компьютерного моделирования при решении задачи исследования структуры археологических данных на примере созданного набора данных, содержащего археологические памятники бронзового века на территории Южного Зауралья. В заключении представлены результаты применения алгоритма сегментации на основе DGCNN* на примере набора данных ModelNet 40, указано направление дальнейших исследований.

1. Мультимодальный динамический граф DGCNN*

1.1. Определение динамического графа сверточной нейронной сети DGCNN*

Археологические памятники включают в себя множество объектов и артефактов, поэтому метод сегментации должен извлекать как глобальные признаки памятника, так и локальные признаки, которые связаны с конкретными объектами или их частями. Входные данные для DGCNN* представляют плотное нерегулярное облако точек $C = \{c_1, ..., c_n\}$ в $\mathbb{R}^3$, где $i = 1, ..., n$, $c_i = (c_i^x, c_i^y, c_i^z, n_i^x, n_i^y, n_i^z, R_i, G_i, B_i)^\intercal$ — вектор признаков точки в виде координат точки $(c_i^x, c_i^y, c_i^z)^\intercal$, нормалей $(n_i^x, n_i^y, n_i^z)^\intercal$ и компонент цвета $(R_i, G_i, B_i)^\intercal$. Пусть m — число семантических меток $L = \{l_1, ..., l_k\}$ в $\mathbb{N}$, где $k = 1, ..., n$, тогда выход сети DGCNN* $C = \{c_1^S, ..., c_n^S\}$ будет иметь размерность $n \times m$ для каждой точки в облаке C. Каналы цветовой модели тон, насыщенность, значение (HSV, hue, saturation, value) обладают различной типологической информацией, поэтому их можно рассматривать как независимые признаки в отличие от каналов цветовой модели красный, зеленый, синий (RGB, red, green, blue). Поэтому в работе признаки RGB преобразованы в HSV. DGCNN* основана на концепции построения динамического графа, формируемого путем пересчета матрицы Кирхгофа графа в каждом сверточном слое сети. DGCNN* принимает плотные облака точек с выбранным набором признаков, анализирует локальные особенности объектов с использованием множественной операции свертки на графах в специальных слоях EdgeConv, и затем с помощью метрического классификатора на базе двух многослойных сетей прямого распространения (MLP, multi-layer perceptron) и одной радиально-базисной сети (RBF, radial basis function) осуществляет сегментацию 3D объектов.

1.2. Регистрация 3D данных

В археологии дешифрирование археологического памятника осуществляется на основе 3D моделей рельефа, полученных с разных точек обзора. Поэтому возникает необходимость решения задачи реконструкции 3D модели памятника. Введем следующие определения: $X = \{x_1, ..., x_n\}$ и $Y = \{y_1, ..., y_m\}$ — исходное и целевое облако точек в $\mathbf{R}^3$. Одним из известных решений задачи является итеративный алгоритм ближайших точек (ICP, iterative closest point). На основе ICP авторами работы разработан точный комбинированный алгоритм регистрации 3D данных (FICP, fusion iterative closest point) [12], в котором решение вариационной задачи представлено как

$$J(RV, RD) = \frac{\alpha}{\mathrm{W} \mid \mathrm{A_f}\mid} \sum_{i \in A_f}^{m} \left\| M\left(RVF_x^i\right) - M\left(F_y^i\right) \right\|^2 + \frac{(1-\alpha)}{\mathrm{w}\,|A_\mathrm{d}|} \sum_{j \in A_d}^{n} \|RDx_j + T - y_j\|^2, \quad (1)$$

где F_x^i, F_y^i — особые точки [13]; RV — матрица аффинного преобразования для данных о цветовых признаках кадра; RD и T — матрица поворота и вектор переноса для карты глубины соответственно; M — функция преобразования координат особых точек в систему координат камеры; α,W — весовые коэффициенты; A_f — связи между особыми точками; A_d — связи между соответствующими точками x_j и y_j в облаках. Алгоритм позволяет решить проблему зависимости решения вариационной задачи от правильности выбора начальных значений, используется для точной регистрации облаков точек с произвольным пространственным разрешением и масштабом относительно друг друга.

1.3. Предварительная обработка данных

Большинство 3D датчиков глубины генерируют разреженные, зашумленные и неоднородные облака точек, что оказывает негативное влияние на процесс классификации и сегментации 3D объектов. Поэтому для предобработки в DGCNN* использован алгоритм с контролируемой повышающей дискретизацией облака точек на основе метода k ближайших соседей (kNN), который позволяет получить плотное, полное и однородное облако точек. Функция потерь алгоритма

$$L_{up} = \sum_{i=0}^{N_2} \sum_{j \in K(i)} \mu\left(\|c_j - c_i\|\right) \gamma\left(\|c_j - c_i\|\right), \tag{2}$$

где N_2 — количество точек в облаке, $K(i)$ — число k соседей для точки c_i, $\mu(r) = r$ — потеря отталкивания, $\gamma(r) = exp(-\delta r^2)$, δ — гиперпараметр функции быстрого снижения веса γ.

1.4. Построение мультимодального динамического графа

Пусть задан неориентированный граф вида $G = \{P, E, A\}$, где P — множество вершин, E — множество ребер, A — взвешенная симметричная матрица смежности с элементом $(a_{i,j})_{n \times n}$ — вес ребра (i, j), при этом $a_{i,j} \geqslant 0$. Тогда из матрицы A можно получить матрицу Кирхгофа с помощью формулы $L_c := D - A$, где D — матрица степеней с элементом $(d_{i,j})_{n \times n} = \sum_{j=1}^{n} a_{i,j}$. Для удаления вершины графа с большим весов в матрице Кирхгофа необходимо выполнить нормализацию компонентов матрицы $L_c^{\text{sum}} = \mathrm{I} - (\mathrm{D}^+)^{1/2} A (\mathrm{D}^+)^{1/2}$, где D^+ — обратная матрица Мура—Пенроуза, I — единичная матрица, при этом изолированные вершины исключаются из процесса нормализации. Для каждой вершины графа G выполняется установка связи с вектором ее признаков c_i с характеристиками i-ой точки в облаке, данному вектору с помощью функции s_i (сигнал графа) ставится соответствие вида $s \to \mathbb{R}$. Для создания мультимодального динамического графа G используется алгоритм:

Шаг 1. Определение связей для каждой точки облака точек c_i с другими точками c_j в облаке.

Шаг 2. Определение значений веса ребра

$a_{ij} = \exp -\alpha \left(w_1 \left\| c_i^{xyz} - c_j^{xyz} \right\|^2 + w_2 \| n_i^{xyz} - \ n_j^{xyz} \|^2 + w_3 \| \ c_i^{rgb} - c_j^{rgb} \|^2 \right)$, где α — гиперпараметр, управляющий балансом между точностью решения и гладкостью графа, w_1, w_2, w_3 — веса групп признаков.

Шаг 3. Стоп.

1.5. Фильтрация и свертка графа

Облако точек представляет собой нерегулярный набор данных с неупорядоченными вершинами c_i, поэтому возникают проблемы с подбором ядра свертки в области вершин. В работе произведена фильтрация векторов признаков c_i в спектральной области с помощью преобразования Фурье графа, а затем применена аппроксимация с помощью многочлена Чебышёва для повышения производительности процедуры фильтрации и свертки графа. Матрица Кирхгофа имеет собственные значения λ_l, такие что $0 = \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_N$, которым соответствуют собственные векторы $\left\{ \mu^{(l)} = \left(\mu_1^{(l)}, \mu_2^{(l)}, \ldots, \mu_N^{(l)} \right)^{\intercal} \right\}_{l=1}^{N}$. Собственные векторы образуют ортонормированный базис $U = \left(\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(N)} \right)$. Тогда матрица Кирхгофа $L = U \operatorname{diag}(\lambda) U^{\intercal}$, где $\operatorname{diag}(\lambda)$ — диагональная матрица. Прямое преоб-

разование Фурье графа определяется по формуле $GF[s](\lambda_l) = \hat{s}(\lambda_l) = \sum_{i=1}^{N} s(i)\left(\mu_i^{(l)}\right)^{\intercal}$, а обратное — по формуле $\mathrm{IGF}[\hat{s}](i) = s(i) = \sum_{l=0}^{N-1} \hat{s}(\lambda_l)\,\mu_i^{(l)}$. Пусть спектральный фильтр задан вектором $\psi \in \mathbb{R}^N$, тогда компонента вектора ψ_i может быть определена как значение функции $g_\psi : \mathbb{R}_+ \to \mathbb{R}$ в i-ом собственном числе $\psi_i = g_\psi(\lambda_i)$. Пусть на графе заданы G два сигнала s_1 и s_2, тогда, используя свойства преобразования Фурье, получим $s_1 * s_2 = \mathrm{IGF}\left[GF([s_1]) \cdot GF([s_2])\right]$, где $\cdot$ — операция покомпонентного умножения, $*$ — операция свертки. На основе теоремы о свертке функций получим формулы для вычисления свертки двух сигналов на графе

$$\begin{aligned} (s_1 * s_2)(i) &= \sum_{l=0}^{N-1} \widehat{s_1}(\lambda_l)\,\widehat{s_2}(\lambda_l)\,\mu_i^{(l)}, \\ s_2 = g_\psi(L_c^{\mathrm{sum}})\,s_1 &= g_\psi(U\,\mathrm{diag}(\lambda)U^{\intercal})\,s_1. \end{aligned} \tag{3}$$

Для аппроксимации спектральной фильтрации были использованы многочлены Чебышёва, которые позволяют повысить производительность метода сегментации. Пусть N — число коэффициентов Чебышёва в многочлене, тогда запишем функцию g_ψ в виде $g_\psi(x) = \frac{1}{2}x_0 + \sum_{m=1}^{\infty} x_m \bar{T}_m(x), \forall x \in [0, \lambda_N]$. Тогда результат действия n-го многочлена Чебышёва на сигнал s_1 может быть определен по формуле

$$s_2 = g_\psi(L_c^{sum})s_1 \approx \sum_{m=0}^{M-1} \psi_m T_m(L_c^{sum})s_1. \tag{4}$$

При $N = 1$ фильтрация с помощью многочлена Чебышёва аналогична работе однослойного персептрона. Известно, что информация о локальных особенностях не теряется при свертке графа с многочленами Чебышёва высокого порядка $N > 3$. Нормализованная матрица Кирхгофа L_c^{sum} разрежена и операция ее умножения на вектор имеет линейную сложность $O(|E|)$. Тогда для первых N степеней многочлена вычислительная сложность операции свертки сигналов s_1 и s_2 с использованием многочлена Чебышёва равна $O(N|E|)$.

1.6. Функция потерь

Функция потерь L_{los} имеет два слагаемых и основана на вычислении мультиклассовой кросс-энтропии L_{CE}, также в нее добавлен параметр регуляризации L_s, связанный с гладкостью сигнала графа s, который вычисляется по трем слоям свертки DGCNN*. Данный параметр позволяет делать объекты смежных вершин более похожими, что облегчает задачу семантической сегментации. Пусть X_i — входной вектор сигнала, Y_i — целевой вектор сигнала, тогда

$$L_{los}(X_i, Y_i) = L_{CE} + L_s = -\sum_{j=1}^{M} y_{i,j} \log p_{i,j} + \alpha \sum_{k=1}^{3} fm_k^{\intercal} L_c^{sum} fm_k, \tag{5}$$

где M — количество возможных меток класса; $y_{i,j}$ — бинарный признак (маска класса), $p_{i,j}$ — прогнозируемая вероятность модели, которая определяет вероятность того, что (i, j) маска принадлежит классу j; fm_k — карта признаков для k слоя в графе свертки; α — гиперпараметр, в работе используем значение равное 10^{-9}. Предложенная функция потерь L_{los} обеспечивает сглаживание Лапласа в пространственной и спектральной областях.

2. Алгоритм сегментации 3D данных

Входными данными алгоритма сегментации на основе DGCNN* является набор данных из 9 элементов для каждой точки: координаты, координаты вектора нормалей и данные о цвете, в процессе работы алгоритма данный набор дополняется нормализованными координатами. Координаты точки позволяют осуществлять глобальное позиционирование точек во всем облаке точек, в то время как нормализованные координаты представляют расположение внутри локального блока данных в облаке точек.

В работе произведена модификация первого сверточного слоя GCNN таким образом, чтобы kNN мог динамически использовать информацию о нормализованных координатах и нормалях векторов при поиске соседей для каждой точки. Далее информация в сети последовательно обрабатывается тремя слоями EdgeConv с коэффициентами Чебышёва, равными 6, 5 и 3 соответственно, двумя слоями MLP и одним слоем с радиально-базисными функциями с максимальным объединением объектов из разных слоев, что позволяет извлекать признаки глобальных и локальных объектов в облаке точек. Каждый слой EdgeConv включает в себя построение мультимодального динамического взвешенного графа, фильтрацию объектов и свертку графа. Эксперименты показали, что для повышения точности сегментации нужно осуществлять пересчет матрицы Кирхгофа графа G для каждого слоя EdgeConv. С помощью слоев EdgeConv и слоя пулинга на выходе нейронной сети получаем исходный набор данных с оценками сегментации для каждого класса. На вход модуля построения графов подаются только нормализованные координаты, а на вход сверточного слоя подаются как исходные, так и нормализованные координаты точек.

Далее представим алгоритм сегментации 3D моделей данных на основе DGCNN*.

procedure DGCNN*
Входные данные: $C = \{c_1, ..., c_n\}$
Выходные данные: $C = \{c_1^S, \ldots, c_n^S\}$
1: Инициализация: Чебышёв {6,5,3}, k =1 (Счетчик Чебышёва);
2: Дискретизация облака точек (UpSampling);
3: Преобразование RGB $\rightarrow$ HSV; Формирование вектора признаков $\{C_i\}$;
4: **for** k in 1,. . . , 3 **do**
5: Расчет матрицы Кирхгофа L_k;
6: Нормализация компонентов матрицы L^{sum};
7: Апроксимация сигнала графа многочленом Чебышёва g_ψ со степенью k;
8: Свертка графа GCNN: $s_2 = g_\psi(L^{sum})s_1$;
9: Формирование признаков динамического графа G;
10: **end**;
11: MLP 1 (1024);
12: MLP 2 (512);
13: concatenation (EdgeConv 2, MLP2);
14: Формирование результата сегментации на основе RBF;
15: **Вычисление функции потерь** $L_{los}(X_i, Y_i)$ (см. (5)).

На рис. 1 представлена архитектура сверточной нейронной сети на основе регулярного динамического взвешенного графа DGCNN*.

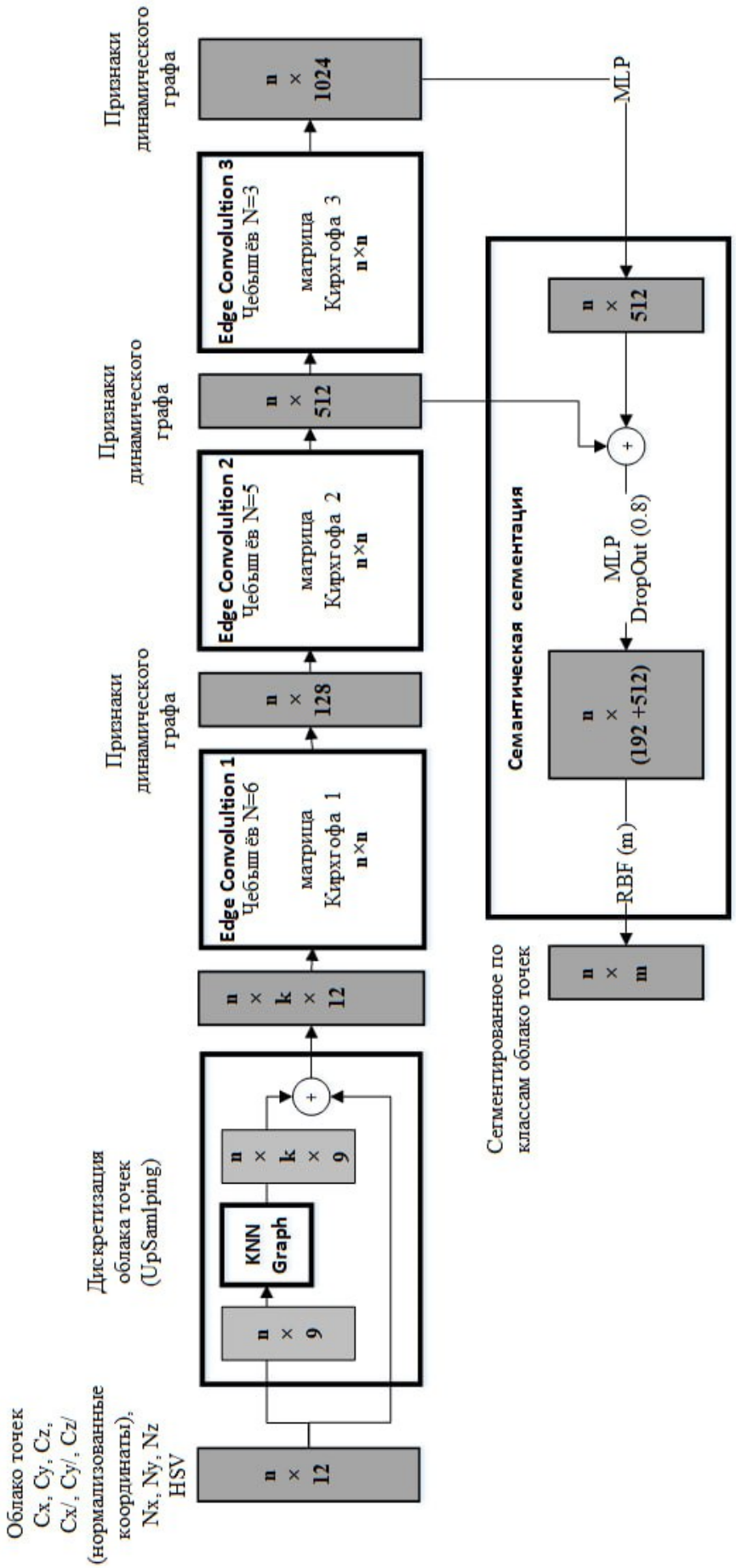


Рис. 1. Архитектура сверточной нейронной сети DGCNN*

3. Компьютерное моделирование

В работе проведено компьютерное моделирование с использованием эталонного набора ModelNet 40 (табл. 2 и табл. 3) и набора данных, содержащего археологические памятники бронзового века на территории Южного Зауралья [14], а именно облаков точек, полученных в результате тахеометрической съемки археологического комплекса Синташта (табл. 4 и табл. 5) с использованием тахеометра Trimble 3300. Для компьютерного моделирования на коллекции данных, содержащей археологические памятники бронзового века, для регистрации облаков точек были использованы алгоритмы итеративного алгоритма ближайших точек на основе нормалей (NICP, Normal Iterative Closest Point) и комбинированного итеративного алгоритма ближайших точек (FICP).

Таблица 2. Точность 3D сегментации для эталонного набора данных ModelNet 40 для исходного облака точек

Наименование метода	Mean Acc Обучающая выборка	Mean Acc Валид. выборка	Mean Acc Тестовая выборка	F1-мера
Point Net	0.683	0.533	0.520	0.247
Point Net++	0.734	0.648	0.611	0.412
DGCNN	0.891	0.796	0.744	0.747
DGCNN+RGB	0.902	0.841	0.817	0.821
RGCNN	0.896	0.855	0.819	0.789
ConvPoint	0.916	0.903	0.894	0.833
KPConv	0.941	0.886	0.861	0.866
DGCNN*	0.934	0.923	0.907	0.854

Таблица 3. Точность 3D сегментации для эталонного набора данных ModelNet 40 для выровненного облака точек

Наименование метода	Mean Acc Обучающая выборка	Mean Acc Валид. выборка	Mean Acc Тестовая выборка	F1-мера
DGCNN* + NICP	0.712	0.686	0.674	0.444
DGCNN* + FICP	0.825	0.798	0.776	0.631

Во втором случае облако точек не содержало данных о цвете, поэтому была проведена его раскраска с использованием системы картографирования Trimble MX9 и соответствующих RGB кадров. Из ModelNet 40 было отобрано четыре набора данных с общим количеством объектов 456. Облака точек, полученные с помощью тахеометрической съемки, содержат от 400 до 1200 точек для каждого археологического бронзового памятника, поэтому требуется увеличение плотности облака точек.

Таблица 4. Точность 3D сегментации по данным тахеометрической съемки для исходного облака точек

Наименование метода	Mean Acc Обучающая выборка	Mean Acc Валид. выборка	Mean Acc Тестовая выборка	F1-мера
Point Net	0.623	0.476	0.471	0.256
Point Net++	0.655	0.644	0.602	0.398
DGCNN	0.792	0.713	0.688	0.646
DGCNN+RGB	0.776	0.708	0.653	0.629
ConvPoint	0.833	0.8	0.711	0.625
KPConv	0.796	0.746	0.735	0.634
RGCNN	0.814	0.776	0.72	0.725
DGCNN*	0.862	0.791	0.773	0.622

Таблица 5. Точность 3D сегментации по данным тахеометрической съемки для выровненного облака точек

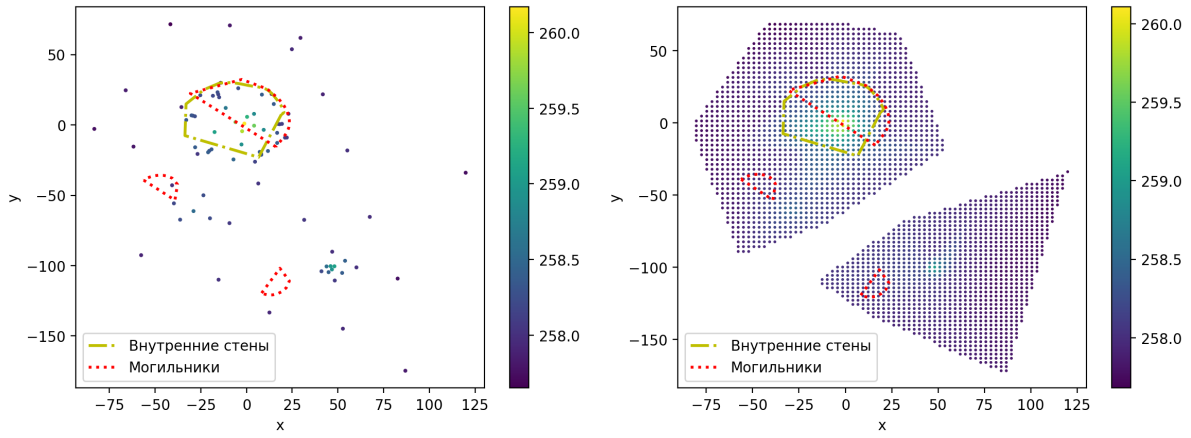
Наименование метода	Mean Acc Обучающая выборка	Mean Acc Валид. выборка	Mean Acc Тестовая выборка	F1-мера
DGCNN* + NICP	0.69	0.543	0.511	0.475
DGCNN* + FICP	0.786	0.742	0.753	0.685

Далее представим псевдокод модифицированного иерархического алгоритма агломеративной кластеризации.

procedure GLA*
Входные данные: max distance= 0.3, $P^{IN} = \{p_1^{IN}, ..., p_n^{IN}\}$
Выходные данные: $P^{OUT} = \{p_1^{OUT}, ..., p_m^{OUT}\}$
$points_{norm}$ = Нормализация точек P^{IN} в интервале [0...1];
$model_{cluster}$ = new Agglomerative Clustering (max distance);
$model_{cluster}$ = Обучение нейронной сети ($points_{norm}$);
$Clusters = \{c_1, ..., c_n\} = model_{cluster}$. Получить кластеры($points_{norm}$);
for Кластер$_j$ in $model_{cluster}$.Кластеры() **do**
    $points_{cluster}$ = Фильтрация ($points_{norm}$, Кластер$_j$);
    if $points_{cluster}$.Счетчик точек() $< p_{aug}$ **then** // $p_{aug} = 4$, параметр кластеризации
        P^{OUT}.Добавление точек($points_{cluster}$);
        continue;
    end;
    $Grid_{points}$ = Создать сетку точек (Крайние точки. $points_{cluster}$);
    $points_{interpolation}$ = Интерполяция ($points_{cluster}, Grid_{points}$);
    P^{OUT}.($points_{interpolation}$);
end;
Формирование результата агломеративной кластеризации.

Для увеличения плотности облака точек и повышения степени равномерности точек в облаке в работе были использованы модифицированный иерархический алгоритм агломеративной кластеризации Agglomerative Clustering, а точнее вариант на основе GLA* (Generic_linkage algorithm) [15] и алгоритм повышающей дискретизации облаков точек на основе графовой сверточной нейронной сети (PU-GCN, point cloud upsampling using graph convolutional networks) [16].

На рис. 2 представлены результаты работы алгоритма GLA* для могильника вблизи п. Осиповка.



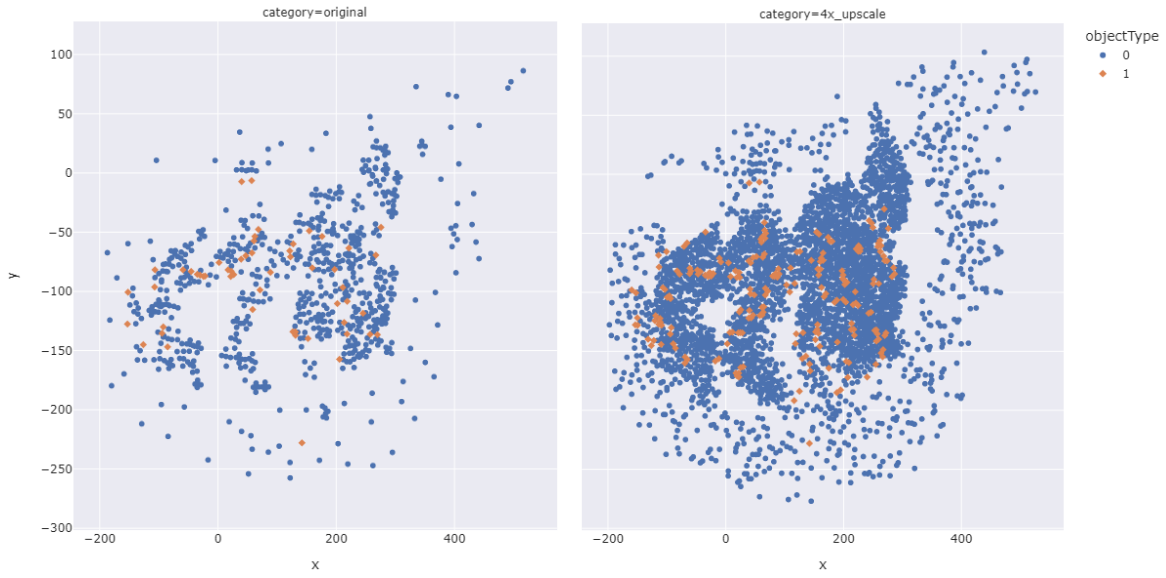
(a) (b)

Рис. 2. Результаты повышения размерности облака точек с использованием алгоритма GLA* на примере могильника вблизи п. Осиповка: a) облако точек до применения алгоритма агломеративной кластеризации b) облако точек после применения алгоритма агломеративной кластеризации

На рис. 3 представлены результаты работы алгоритма повышающей дискретизации облаков точек PU-GCN на примере археологического памятника бронзового века вблизи п. Каменка Челябинской области.

Построенные модели обучены на ПК на базе Intel Core i7 с картой Nvidia GeForce GTX 1080Ti в течение 150 эпох. Мы провели дискретизацию облака точек на основе kNN с $k = 20$. Был проведен сравнительный анализ предложенного метода сегментации с известными методами 3D сегментации и разными комбинациями входных признаков облаков точек, исследовано влияние на точность сегментации способа формирования облака точек: в первом случае исследовалось облако точек из эталонного набора данных (табл. 2 и табл. 4), во втором случае применены варианты с использованием 3D регистрации на основе алгоритмов NICP И FICP (табл. 3 и табл. 5).

Моделирование показало превосходство предложенного метода по всем метрикам над Point Net, Point Net++ и DGCNN, для RGCNN и DGCNN+RGB получены близкие по точности результаты по метрике F1 и лучшие результаты по метрике Mean Acc. Использование методов регистрации для формирования облака точек ожидаемо приводит к уменьшению точности процедуры его сегментации, однако метод FICP [17] имеет преимущества по сравнению с NICP.



(a) (b)

Рис. 3. Результаты повышения плотности облака точек с использованием алгоритма PU-GCN на примере археологического памятника вблизи п. Каменка: a) облако точек до применения алгоритма повышающей дискретизации b) облако точек после применения алгоритма повышающей дискретизации (класс 0 — «фон», класс 1 — «жилищная впадина»)

Заключение

В работе предложен метод семантической сегментации нерегулярных облаков точек на основе мультимодального взвешенного динамического графа DGCNN*. Архитектура сети использует принцип динамического определения соседства точек в облаке на основе данных о геометрии облаков точек и данных о цвете, что позволяет устранить главный недостаток DGCNN и RGCNN, связанный с размерностью обрабатываемых облаков точек. Структура полносвязного слоя в виде метрического классификатора из двух MLP сетей и RFB сети с конкатенацией данных, получаемых с EdgeConv слоев сети, позволяет эффективно обрабатывать локальные и глобальные признаки объектов. Предложенный метод позволяет получать точное решение задачи семантической сегментации для разреженных, зашумленных и неоднородных облака точек, для 3D сцен с микрорельефом и объектами невыпуклой формы, что важно при обработке геопространственных данных. Метод независим от способа сбора данных: показывает достаточно высокую точность для облаков точек, полученных с использованием вариантов алгоритма ICP.

Исследование выполнено за счет гранта РНФ (проект 23-11-20007).

Литература

1. Su H., Maji S., Kalogerakis E., Learned–Miller E. Multi-view Convolutional Neural Networks for 3D Shape Recognition // IEEE Proceedings of International Conference on Computer Vision (ICCV) (Santiago, Chile, December 7–13, 2015). P. 945–953. DOI: 10.1109/ICCV.2015.114.

2. Charles R.Q., Su H., Kaichun M., Guibas L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Honolulu, HI, USA, July 21–26, 2017). P. 77–85. DOI: 10.1109/CVPR.2017.16.

3. Charles R.Q., Li Y., Hao S., Leonidas J.G. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space // Proceedings of 31st Conference on Neural Information Processing Systems (NIPS) (Long Beach, USA, December 4–9, 2017). P. 5099–5108. DOI: 10.48550/arXiv.1706.02413.

4. Zhang Y., Rabbat M. A Graph-CNN for 3D Point Cloud Classification // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Calgary, AB, Canada, April 15–20, 2018). P. 6279–6283. DOI: 10.1109/ICASSP.2018.8462291.

5. Wang Y., Sun. Y., Liu Z., *et al.* Dynamic Graph CNN for Learning on Point Clouds // ACM Transactions on Graphics. 2019. Vol.38, No. 5. Article 146. P. 1–12. DOI: 10.1145/3326362.

6. Te G., Hu W., Zheng A., Guo Z. RGCNN: Regularized Graph CNN for Point Cloud Segmentation // Proceedings of the 26th ACM international conference on Multimedia (MM '18) (Seoul, Republic of Korea, October 22–26, 2018). ACM, 2018. P. 746–754. DOI: 10.1145/3240508.3240621.

7. Boulch A. ConvPoint: Continuous convolutions for point cloud processing // Computers and Graphics. 2020. Vol. 88. P. 24–34. DOI: 10.1016/j.cag.2020.02.005.

8. Hugues T., Charles R.Q., Deschaud J.E., *et al.* KPConv: Flexible and Deformable Convolution for Point Clouds // IEEE/CVF International Conference on Computer Vision (ICCV) (Seoul, Republic of Korea, October 27 – November 2, 2019). P. 6410–6419. DOI: 10.1109/ICCV.2019.00651.

9. Zhang Z., Hua B.S., Yeung S.K. ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics // IEEE/CVF International Conference on Computer Vision (ICCV) (Seoul, Republic of Korea, October 27 – November 2, 2019). P. 1607–1616. DOI: 10.1109/ICCV.2019.00169.

10. Damien R., Hugo R., Loic L. Efficient 3D semantic segmentation with superpoint transformer // IEEE/CVF International Conference on Computer Vision (ICCV) (Paris, France, October 1–6, 2023). P. 17149–17158. DOI: 10.1109/ICCV51070.2023.0157.

11. 3D Semantic Segmentation on DALES. URL: https://paperswithcode.com/sota/3d-semantic-segmentation-on-dales (дата обращения: 31.03.2024).

12. Вохминцев А.В., Мельников А.В., Пачганов С.А. Метод навигации и составления карты в трехмерном пространстве на основе комбинированного решения вариационной подзадачи точка-точка ICP для аффинных преобразований // Информатика и ее применения. 2020. Т. 14, № 1. С. 101–112. DOI: 10.14357/19922264200114.

13. Вохминцев А.В, Соченков И.В., Кузнецов В.В., Тихоньких Д.В. Распознавание лиц на основе алгоритма сопоставления изображений с рекурсивным вычислением гистограмм направленных градиентов // Доклады академии наук. 2016. Т. 93, № 1. С. 37–41. DOI: 10.1134/S1064562416010178.

14. Vokhmintcev A.V., Khristodulo O.I, Melnikov A.V., Romanov M.A. Application of Dynamic Graph CNN* and FICP for Detection and Research Archaeology Sites // Analysis of Images,

Social Networks and Texts (AIST 2023). Vol. 14486 / eds. by D.I. Ignatov, *et al.* Cham: Springer, 2024. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-54534-4_21.

15. Day W.H.E., Edelsbrunner H. Efficient algorithms for agglomerative hierarchical clustering methods // Journal of Classification. 1984. Vol. 1, no. 1. P. 7–24. DOI: 10.1007/BF01890115.
16. Qian G., Abualshour A., Li G., *et al.* PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (Nashville, TN, USA, June 20–25, 2021). DOI: 10.1109/CVPR46437.2021.01151.
17. Vokhmintsev A.V., Khristodulo O.I., Romanov M.A. Semantic Classification and Segmentation of Archaeological Sites Based on a Fusion of Object Detector and 3DEF // 2023 International Russian Automation Conference (RusAutoCon) (Sochi, Russian Federation, September 10–16, 2023). P. 122–127. DOI: 10.1109/RusAutoCon58002.2023.10272916.

Вохминцев Александр Владиславович, д.т.н., заведующий, научно-исследовательская лаборатория интеллектуальных информационных технологий и систем, Челябинский государственный университет (Челябинск, Российская Федерация); профессор, Югорский государственный университет (Ханты-Мансийск, Российская Федерация)

Аббазов Валериан Ринатович, лаборант-исследователь, научно-исследовательская лаборатория интеллектуальных информационных технологий и систем, Челябинский государственный университет (Челябинск, Российская Федерация)

Романов Матвей Александрович, лаборант-исследователь, научно-исследовательская лаборатория интеллектуальных информационных технологий и систем, Челябинский государственный университет (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240202

MULTIMODAL DYNAMIC GRAPH CNN FOR 3D SEMANTIC SEGMENTATION

© 2024 A.V. Vokhmintcev[1,2], V.R. Abbazov[1], M.A. Romanov[1]

[1] *Chelyabinsk State University (st. Brat'yev Kashirinykh 129, Chelyabinsk, 454001 Russia),*

[2] *Yugra State University (st. Chekhova 16, Khanty-Mansiysk, 628012 Russia)*

E-mail: vav@csu.ru, abbavar@yandex.ru, std.romanov.ma@gmail.com

Received: 01.04.2024

In this paper, a semantic segmentation method of point clouds in the form of terrain using a new multimodal convolutional neural network architecture based on a regular dynamic weighted graph, which allows to obtain an accurate solution to the segmentation problem based on a fusion of geometric and color features. The method can be effectively used for sparse, noisy, inhomogeneous and non-convex point clouds. The computer modeling of state-of-the-art methods for 3D semantic segmentation was carried out using the reference data collection ModelNet 40 and a data set of archaeological sites of the Bronze Age of the Southern Trans-Urals, namely data obtained as a result of a total station survey (the Trimble 3300 total station) of a complex of archaeological sites in the valley of the Sintashta river. A comparative analysis of the proposed method and state-of-the-art methods for 3D semantic segmentation with different combinations of input features of point clouds was carried out, and the method influence of forming a point cloud on the accuracy of 3D semantic segmentation was also investigated: in the first case, a point cloud from a reference dataset was studied, in the second case, variants using 3D registration based on NICP and FICP algorithms were applied.

Keywords: segmentation of 3D objects, graph convolutional neural networks, point clouds registration.

FOR CITATION

Vokhmintcev A.V., Abbazov V.R., Romanov M.A. Multimodal Dynamic Graph CNN for 3D Semantic Segmentation. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 23–38. (in Russian) DOI: 10.14529/cmse240202.

References

1. Su H., Maji S., Kalogerakis E., Learned–Miller E. Multi-view Convolutional Neural Networks for 3D Shape Recognition. IEEE Proceedings of International Conference on Computer Vision (ICCV) (Santiago, Chile, December 7–13, 2015). P. 945–953. DOI: 10.1109/ICCV.2015.114.

2. Charles R.Q., Su H., Kaichun M., Guibas L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Honolulu, HI, USA, July 21–26, 2017). P. 77–85. DOI: 10.1109/CVPR.2017.16.

3. Charles R.Q., Li Y., Hao S., Leonidas J.G. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. Proceedings of 31st Conference on Neural Information Processing Systems (NIPS) (Long Beach, USA, December 4–9, 2017). P. 5099–5108. DOI: 10.48550/arXiv.1706.02413.

4. Zhang Y., Rabbat M. A Graph-CNN for 3D Point Cloud Classification. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Calgary, AB, Canada, April 15–20, 2018). P. 6279–6283. DOI: 10.1109/ICASSP.2018.8462291.

5. Wang Y., Sun. Y., Liu Z., *et al.* Dynamic Graph CNN for Learning on Point Clouds. ACM Transactions on Graphics. 2019. Vol. 38. no. 5. Article 146. P. 1–12. DOI: 10.1145/3326362.

6. Te G., Hu W., Zheng A., Guo Z. RGCNN: Regularized Graph CNN for Point Cloud Segmentation. Proceedings of the 26th ACM international conference on Multimedia (MM '18) (Seoul, Republic of Korea, October 22–26, 2018). ACM, 2018. P. 746–754. DOI: 10.1145/3240508.3240621.

7. Boulch A. ConvPoint: Continuous convolutions for point cloud processing. Computers and Graphics. 2020. Vol. 88. P. 24–34. DOI: 10.1016/j.cag.2020.02.005.

8. Hugues T., Charles R.Q., Deschaud J.E., *et al.* KPConv: Flexible and Deformable Convolution for Point Clouds. IEEE/CVF International Conference on Computer Vision (ICCV) (Seoul, Republic of Korea, October 27 – November 2, 2019). P. 6410–6419. DOI: 10.1109/ICCV.2019.00651.

9. Zhang Z., Hua B.S., Yeung S.K. ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics. IEEE/CVF International Conference on Computer Vision (ICCV) (Seoul, Republic of Korea, October 27 – November 2, 2019). P. 1607–1616. DOI: 10.1109/ICCV.2019.00169.

10. Damien R., Hugo R., Loic L. Efficient 3D semantic segmentation with superpoint transformer. IEEE/CVF International Conference on Computer Vision (ICCV) (Paris, France, October 1–6, 2023). P. 17149–17158. DOI: 10.1109/ICCV51070.2023.0157.

11. 3D Semantic Segmentation on DALES. URL: https://paperswithcode.com/sota/3d-semantic-segmentation-on-dales (accessed: 31.03.2024).

12. Vokhmintcev A.V., Melnikov A.V., Pachganov S.A. Simultaneous localization and mapping method in three-dimensional space based on the combined solution of the point-point variation problem icp for an affine transformation. Informatics and Applications. 2020. Vol. 14, no. 1. P. 101–112. DOI: 10.14357/19922264200114.

13. Vokhmintcev A.V., Sochenkov I.V., Kuznetsov V.V., Tikhonkikh D.V. Face recognition based on matching algorithm with recursive calculation of local oriented gradient histogram. Doklady Mathematics. 2016. Vol. 93, no. 1. P. 37–41. DOI: 10.1134/S1064562416010178.

14. Vokhmintcev A.V., Khristodulo O.I, Melnikov A.V., Romanov M.A. Application of Dynamic Graph CNN* and FICP for Detection and Research Archaeology Sites. Analysis of Images, Social Networks and Texts (AIST 2023). Vol. 14486 / eds. by D.I. Ignatov, *et al.* Cham: Springer, 2024. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-54534-4_21.

15. Day W.H.E., Edelsbrunner H. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of Classification. 1984. Vol. 1, no. 1. P. 7–24. DOI: 10.1007/BF01890115.

16. Qian G., Abualshour A., Li G., *et al.* PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (Nashville, TN, USA, June 20–25, 2021). DOI: 10.1109/CVPR46437.2021.01151.

17. Vokhmintsev A.V., Khristodulo O.I., Romanov M.A. Semantic Classification and Segmentation of Archaeological Sites Based on a Fusion of Object Detector and 3DEF. 2023 International Russian Automation Conference (RusAutoCon) (Sochi, Russian Federation, September 10–16, 2023). P. 122–127. DOI: 10.1109/RusAutoCon58002.2023.10272916.

УДК 519.254, 004.048 **DOI: 10.14529/cmse240203**

ВОССТАНОВЛЕНИЕ МНОГОМЕРНЫХ ВРЕМЕННЫХ РЯДОВ НА ОСНОВЕ ВЫЯВЛЕНИЯ ПОВЕДЕНЧЕСКИХ ШАБЛОНОВ И ПРИМЕНЕНИЯ АВТОЭНКОДЕРОВ

© 2024 А.А. Юртин

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: iurtinaa@susu.ru

Поступила в редакцию: 01.05.2024

В настоящее время в широком спектре предметных областей актуальной является задача восстановления пропущенных точек или блоков значений временных рядов. В статье представлен метод SAETI (Snippet-based Autoencoder for Time-series Imputation) для восстановления пропусков в многомерных временных рядах, который основан на совместном применении нейросетевых моделей-автоэнкодеров и аналитического поиска во временном ряде поведенческих шаблонов (сниппетов). Восстановление многомерной подпоследовательности, содержащей пропуски, выполняется посредством двух следующих нейросетевых моделей. Распознаватель получает на вход подпоследовательность, в которой пропуски предварительно заменены на нули, и для каждого измерения определяет соответствующий сниппет. Реконструктор принимает на вход подпоследовательность и набор сниппетов, полученных Распознавателем, и заменяет пропуски на правдоподобные синтетические значения. Реконструктор реализован как совокупность двух следующих моделей: Энкодер, формирующий скрытое состояние для совокупности входной подпоследовательности и распознанных сниппетов; Декодер, получающий на вход скрытое состояние, который восстанавливает исходную подпоследовательность. Представлено детальное описание архитектур вышеперечисленных моделей. Результаты экспериментов над реальными временными рядами из различных предметных областей показывают, что SAETI в среднем опережает передовые аналоги по точности восстановления и показывает лучшие результаты в случае, когда восстанавливаются данные, отражающие активность некоего субъекта.

Ключевые слова: временной ряд, восстановление пропущенных значений, автоэнкодер, поведенческие шаблоны (сниппеты) временного ряда, нейронные сети.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Юртин А.А. Восстановление многомерных временных рядов на основе выявления поведенческих шаблонов и применения автоэнкодеров // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 39–55. DOI: 10.14529/cmse240203.

Введение

В настоящее время в широком спектре приложений возникает задача обработки временных рядов, содержащих пропущенные значения ввиду аппаратно-программных сбоев и человеческого фактора: Интернет вещей [1], управление системами жизнеобеспечения [2], моделирование климата [3] и финансы [4] и др. В подобных приложениях во временных рядах требуется заменить пропуски на синтетические значения, близкие к исходным, чтобы сохранить целостность данных и минимизировать искажения результатов их интеллектуального анализа. Арсенал подходов к решению задачи восстановления пропусков во временных рядах, разработанных научным сообществом, чрезвычайно широк и включает в себя статистические методы [5], аналитические алгоритмы [6, 7] и интенсивно развивающиеся в настоящее время нейросетевые модели [8, 9].

В данной статье представлен новый метод востановления пропущенных значений многомерного временного ряда SAETI (Snippet-based Autoencoder for Time-series Imputation),

основанный на совместном применении поведенческих шаблонов (сниппетов [10]) и нейронных сетей-автоэнкодеров. SAETI включает в себя две последовательно применяемые нейросетевые модели: Распознаватель, определяющий поведенческий шаблон, на который похожа входная подпоследовательность ряда с пропущенными значениями, и Реконструктор, выполняющий восстановление пропусков на основе информации, которая поступила от Распознавателя. Данная работа развивает исследование [11] в следующих аспектах: предложенный метод предназначен для восстановления блоков пропущенных данных временного ряда (а не последнего значения ряда, полученного в режиме реального времени) и применяет автоэнкодеры для реализации Реконструктора, что позволяет эффективно восстанавливать блоки пропущенных данных.

Статья организована следующим образом. Раздел 1 содержит краткий обзор работ по тематике исследования. В разделе 2 приводятся формальные определения понятий и нотация, используемые в статье. Предложенный метод описан в разделе 3. Результаты вычислительных экспериментов, исследующих эффективность предложенного метода, приведены в разделе 4. Заключение содержит сводку полученных результатов и направления будущих исследований.

1. Обзор связанных работ

Разработка моделей, методов и алгоритмов, обеспечивающих как можно более точное восстановление временных рядов из различных предметных областей, является одной из наиболее актуальных задач обработки данных [7]. Для решения указанной задачи на сегодня разработано большое количество как аналитических алгоритмов, так и методов на основе нейросетевых моделей. В настоящее время наиболее эффективными аналитическими алгоритмами признаются [6, 7] следующие разработки: ORBITS [7], DynaMMo [12], CDRec [13], SoftImpute [14] и др. Нейросетевые методы восстановления временных рядов используют широкий спектр современных архитектур нейронных сетей (см., например, обзоры [8, 9]): рекуррентные нейронные сети (например, BRITS [15], M-RNN [16] и др.), генеративно-состязательные сети (например, E^{2}GAN [17], BRNN-GAN [18] и др.), трансформеры с механизмом самовнимания (self-attention; например, SAITS [19], STING [20]), автоэнкодеры и др. Далее кратко рассмотрены методы NAOMI [21] и GP-VAE [22], которые являются типичными представителями класса методов восстановления временных рядов на основе автоэнкодеров и потому наиболее близки по тематике к исследованию, описанному в данной статье.

Автоэнкодеры представляют собой класс нейронных сетей, используемый для нелинейного снижения размерности входных данных. Структуру автоэнкодера можно разделить на две основные части: энкодер (encoder) и декодер (decoder). Энкодер выполняет процесс сжатия входных данных в скрытое состояние (hidden state) меньшей размерности. Скрытое состояние является точкой в скрытом пространстве (hidden space), содержащей важные характеристики входных данных. Скрытое пространство представляет собой множество всех возможных значений скрытого состояния, которые могут быть сгенерированы энкодером. Каждая точка в таком пространстве соответствует определенному входному образцу, преобразованному в скрытое состояние. Декодер выполняет процесс восстановления исходных данных из скрытого состояния. Во время обучения главной задачей модели является минимизация различия между входными данными и данными, восстановленными из скрытого состояния.

При восстановлении пропущенных значений временных рядов типичной является следующая схема применения автоэнкодеров. Во время предварительной обработки пропуски и часть существующих точек заменяются специальными значениями (например, нулями). Автоэнкодер изучает скрытые закономерности в данных, необходимые для качественного сжатия временного ряда, игнорируя шум, сформированный инициализацией пропущенных значений. В процессе обучения ошибка модели вычисляется между восстановленными декодером значениями и существующими точками, искусственно помеченными как пропущенные. Такой подход позволяет модели обучаться кодировать и декодировать входные данные, игнорируя пропуски.

NAOMI (Non-Autoregressive Multiresolution Sequence Imputation) состоит из двух ключевых компонентов: двунаправленного кодировщика (forward-backward encoder), преобразующего существующие точки временного ряда в скрытые состояния, и многоуровневого декодера, осуществляющего восстановление пропущенных значений на основе доступных скрытых состояний других точек. Декодер проводит процесс восстановления рекурсивно, переходя от максимально удаленных друг от друга точек подпоследовательности к соседним.

GP-VAE (Gaussian Process Variational AutoEncoder) [22] представляет собой модель, использующую для восстановления вариационный автоэнкодер (VAE) [23] и моделирование гауссовского процесса (GP) [24]. В отличие от автоэнкодера, VAE основан на вероятностной модели скрытых переменных. VAE кодирует входные данные в скрытое состояние, представленное в виде параметров многомерного распределения (среднее и дисперсия), определенных в скрытом пространстве. Разработчики данной модели предполагают, что эволюцию скрытых переменных входных подпоследовательностей во времени можно представить как гауссовский процесс (вероятностный процесс, в котором каждая конечная комбинация случайных переменных распределена нормально). Результатом моделирования гауссовского процесса является аппроксимация скрытого состояния входных данных. Аппроксимированное скрытое состояние поступает на вход декодера для формирования выхода модели.

2. Основные определения и нотация

Ниже приводятся обозначения и определения терминов, используемых в данной статье, в соответствии с работой [10].

2.1. Временной ряд и подпоследовательность

Одномерный временной ряд представляет собой хронологически упорядоченную последовательность вещественных значений:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1}$$

Длина временного ряда, n, обозначается как $|T|$.

Подпоследовательность $T_{i,\,m}$ одномерного временного ряда T — это непрерывный промежуток из m элементов ряда, начиная с i-го элемента:

$$T_{i,\,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \leqslant i \leqslant n-m+1, \quad 3 \leqslant m \ll n. \tag{2}$$

Многомерный временной ряд — это набор семантически связанных одномерных временных рядов одинаковой длины, которые синхронизированы во времени. Пусть d обозначает *размерность* многомерного ряда ($d > 1$), количество *измерений* — одномерных рядов в нем.

Подобно одномерному случаю, многомерный временной ряд, его подпоследовательность и отдельные точки обозначим как $\boldsymbol{T}$, $\boldsymbol{T}_{i,m}$ и $\boldsymbol{t}_i$ соответственно, и определим их следующим образом:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{3}$$

$$\boldsymbol{T}_{i,m} = [\{T_{i,m}^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{4}$$

$$\boldsymbol{t}_i = [\{t_i^{(k)}\}_{k=1}^{d}]^{\intercal}. \tag{5}$$

Множество подпоследовательностей многомерного временного ряда будем обозначать как $\boldsymbol{S}_{\boldsymbol{T}}^{m}$:

$$\boldsymbol{S}_{\boldsymbol{T}}^{m} = \{\boldsymbol{T}_{i,m}\}_{i=1}^{n-m+1}. \tag{6}$$

2.2. Сниппеты (поведенческие шаблоны) временного ряда

Сниппеты представляют собой подпоследовательности временного ряда, выражающие типичные активности субъекта, деятельность которого описывает данный ряд, и определяются следующим образом [10].

Для заданной длины подпоследовательности m представим временной ряд T как набор не перекрывающихся подпоследовательностей, *сегментов*. Поскольку $m \ll n$, то без ограничения общности полагаем, что n кратно m, и набор сегментов Seg_T^m определяется следующим образом:

$$Seg_T^m = \{Seg_i\}_{i=1}^{n/m}, \quad Seg_i = T_{m\cdot(i-1)+1,\,m}. \tag{7}$$

Сниппеты T выбираются из Seg_T^m. Введем положительное целое число K ($1 \leqslant K \leqslant n/m$), которое представляет собой ожидаемое количество сниппетов (типичных активностей исследуемого субъекта). Определим, C_T^m, набор сниппетов длины m следующим образом:

$$C_T^m = \{C_i\}_{i=1}^{K}, \quad C_i \in Seg_T^m. \tag{8}$$

Сниппет имеет следующие атрибуты: индекс, набор ближайших соседей и покрытие. Для сниппета $C_i \in C_T^m$ указанные параметры обозначаются как $C_i.index$, $C_i.NN$ и $C_i.frac$ соответственно.

Индекс сниппета представляет собой номер сегмента, которому соответствует сниппет:

$$C_i.index = j \Leftrightarrow Seg_j = T_{m\cdot(j-1)+1,\,m}. \tag{9}$$

Ближайшие соседи сниппета — это набор подпоследовательностей ряда, наименее удаленных от данного сниппета:

$$C_i.NN = \{T_{j,m} \mid Seg_{C_i.index} = \arg\min_{1\leqslant s\leqslant n/m} \mathrm{MPdist}(T_{j,m}, Seg_s), 1 \leqslant j \leqslant n-m+1\}, \tag{10}$$

где $\mathrm{MPdist}(\cdot,\cdot)$ — это функция расстояния, основанная на евклидовой нормированной метрике, предложенная в работе [25]. Покрытие сниппета — это отношение мощности множества его ближайших соседей к общему числу подпоследовательностей ряда соответствующей длины:

$$C_i.frac = \frac{|C_i.NN|}{n-m+1}. \tag{11}$$

В наборе C_T^m сниппеты упорядочиваются в порядке убывания их покрытия:

$$\forall C_i, C_j \in C_T^m : \quad i < j \Leftrightarrow C_i.frac \geqslant C_j.frac. \tag{12}$$

Для многомерного временного ряда $\boldsymbol{T}$ назовем *словарем сниппетов* множество $\boldsymbol{C_T^m}$, объединяющее в себе наборы сниппетов по всем измерениям ряда:

$$\boldsymbol{C_T^m} = \bigcup_{k=1}^{d} C_{T^{(k)}}^m. \tag{13}$$

3. Метод восстановления многомерного временного ряда

3.1. Архитектура

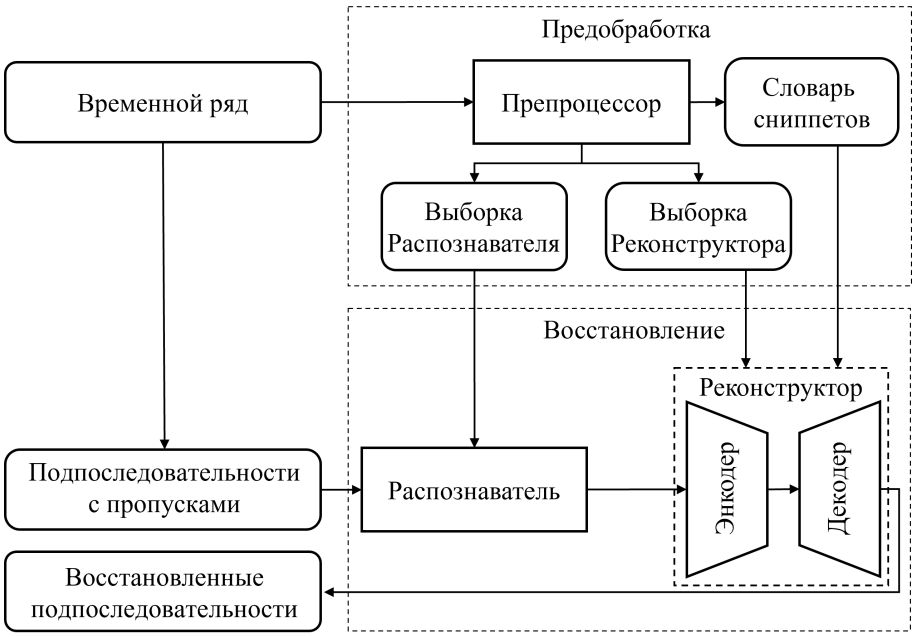


Рис. 1. Метод восстановления многомерного временного ряда

Архитектура предложенного метода представлена на рис. 1 и предполагает подготовку данных и восстановление. На вход поступает многомерный временной ряд, содержащий пропущенные значения. Препроцессор формирует обучающие выборки для нейросетевых моделей метода, Распознавателя и Реконструктора, и словарь сниппетов. Для восстановления подпоследовательности, содержащие пропуски, последовательно обрабатываются Распознавателем и Реконструктором. Распознаватель определяет принадлежность каждого измерения входной многомерной подпоследовательности множеству ближайших соседей сниппетов по данному измерению. Реконструктор, анализируя данные входных подпоследовательностей и сниппеты, заменяет пропуски на синтезированные значения.

3.2. Препроцессор

Препроцессор формирует обучающие выборки Распознавателя и Реконструктора и словарь сниппетов в предположении, что в обрабатываемом временном ряде доля подпоследовательностей, содержащих пропуски, не превышает наперед заданного экспертом в предметной области параметра α, где $0 < \alpha < 1$ и $\alpha = 0.5$ является типичным значением:

$$|\{\, \boldsymbol{T}_{i,m} \in \boldsymbol{S}_{\boldsymbol{T}}^{m} \mid \exists\, t_j \in T_{i,m}^{(k)},\ t_j = \texttt{NaN}\}| \leqslant \alpha \cdot (n - m + 1). \tag{14}$$

На первом шаге предобработки выполняется минимаксная нормализация каждого измерения временного ряда, приводящая значения к диапазону $[0, 1]$:

$$\hat{t}_i = \frac{t_i - \min\limits_{1 \leqslant k \leqslant n} t_k}{\max\limits_{1 \leqslant k \leqslant n} t_k - \min\limits_{1 \leqslant k \leqslant n} t_k}. \tag{15}$$

Затем Препроцессор выполняет поиск сниппетов в каждом измерении ряда $\boldsymbol{T}$, ограничиваясь при этом подпоследовательностями, которые не содержат пропущенные значения. Указанный поиск отличается от оригинального алгоритма поиска сниппетов SnippetFinder [10] и реализован посредством модификации соответствующей функции библиотеки Matrix Profile API [26], что позволило выполнять вычисление профилей расстояний параллельно. Результатом предварительной обработки является словарь сниппетов $\boldsymbol{C}_{\boldsymbol{T}}^{m}$.

Для формального описания формирования обучающих выборок введем оператор, который заменяет на ноль элементы входной многомерной подпоследовательности следующим образом: сначала обнуляются пропущенные значения, затем обнуляется заданная доля оставшихся значений, выбираемых случайным образом. В формальной записи имеем оператор $\texttt{Zero}_\beta\colon \boldsymbol{S}_{\boldsymbol{T}}^{m} \to \boldsymbol{S}_{\boldsymbol{T}}^{m}$, $0 \leqslant \beta \leqslant 1$:

$$\begin{aligned} &\texttt{Zero}_\beta(\boldsymbol{T}_{i,m}) = \hat{\boldsymbol{T}}_{i,m}, \\ &\forall t_j^{(k)} = \texttt{NaN} \quad \hat{t}_j^{(k)} = 0, \\ &\forall t_j^{(k)} \neq \texttt{NaN} \quad \Pr(\hat{t}_j^{(k)} = 0) = \beta. \end{aligned} \tag{16}$$

Применение введенного оператора при формировании обучающих выборок Распознавателя и Реконструктора позволит указанным моделям в процессе обучения адаптироваться и научиться выделять признаки, необходимые для различения случаев, когда данные содержат нули вследствие нормализации, и случаев, когда пропущенные значения заменены на ноль оператором $\texttt{Zero}_\beta$.

Обозначим обучающую выборку нейронной сети как множество пар $D = \langle X, Y \rangle$, где X представляет собой входные данные, а Y — соответствующие им выходные данные. В качестве входных данных обучающей выборки Распознавателю подаются подпоследовательности ряда $\boldsymbol{T}$, для которых на этапе поиска сниппетов была получена разметка. Подпоследовательности не содержат пропущенных значений, однако в них суммарно $\lfloor \beta \cdot d \cdot m \rfloor$ случайных элементов заменены на ноль. Выходными данными полагаются вектора целочисленных значений, где каждый элемент вектора соответствует номеру сниппета, для которого соответствующая координата подпоследовательности входит в множество ближайших соседей сниппета. В итоге формальное определение обучающей выборки Распознавателя выглядит следующим образом:

$$\begin{aligned} D_{\text{Recognizer}} = \{\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \ \mid\ & \boldsymbol{X} = \texttt{Zero}_\beta(\boldsymbol{T}_{i,m}),\ \neg\exists\, t_j^{(k)} = \texttt{NaN} \\ & \boldsymbol{Y} = [\{s^{(k)}\}_{k=1}^{d}]^\intercal,\ T_{i,m}^{(k)} \in C_{s^{(k)}}^{(k)}.NN, \\ & 1 \leqslant i \leqslant n - m + 1,\ 1 \leqslant k \leqslant d,\ 1 \leqslant s \leqslant K\}. \end{aligned} \tag{17}$$

Входными данными обучающей выборки Реконструктора полагаются все подпоследовательности ряда, в которых имеют место пропуски и суммарно $\lfloor \beta \cdot d \cdot m \rfloor$ случайных эле-

ментов заменены на ноль. Выходными данными полагаются подпоследовательности ряда, в которых пропуски были заменены на ноль. В итоге обучающая выборка Реконструктора определяется следующим образом:

$$D_{\text{Reconstructor}} = \{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle \mid \boldsymbol{X} = \texttt{Zero}_{\beta}(\boldsymbol{T}_{i,m}), \quad \boldsymbol{Y} = \texttt{Zero}_{\beta\equiv 0}(\boldsymbol{T}_{i,m}), \quad 1 \leqslant i \leqslant n-m+1,\ 1 \leqslant k \leqslant d\}. \tag{18}$$

3.3. Распознаватель

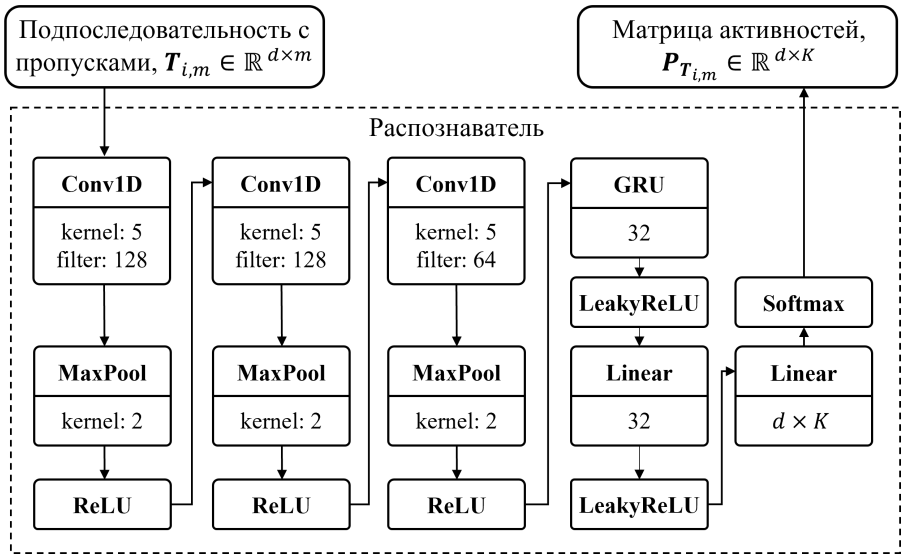


Рис. 2. Структура нейронной сети Распознавателя

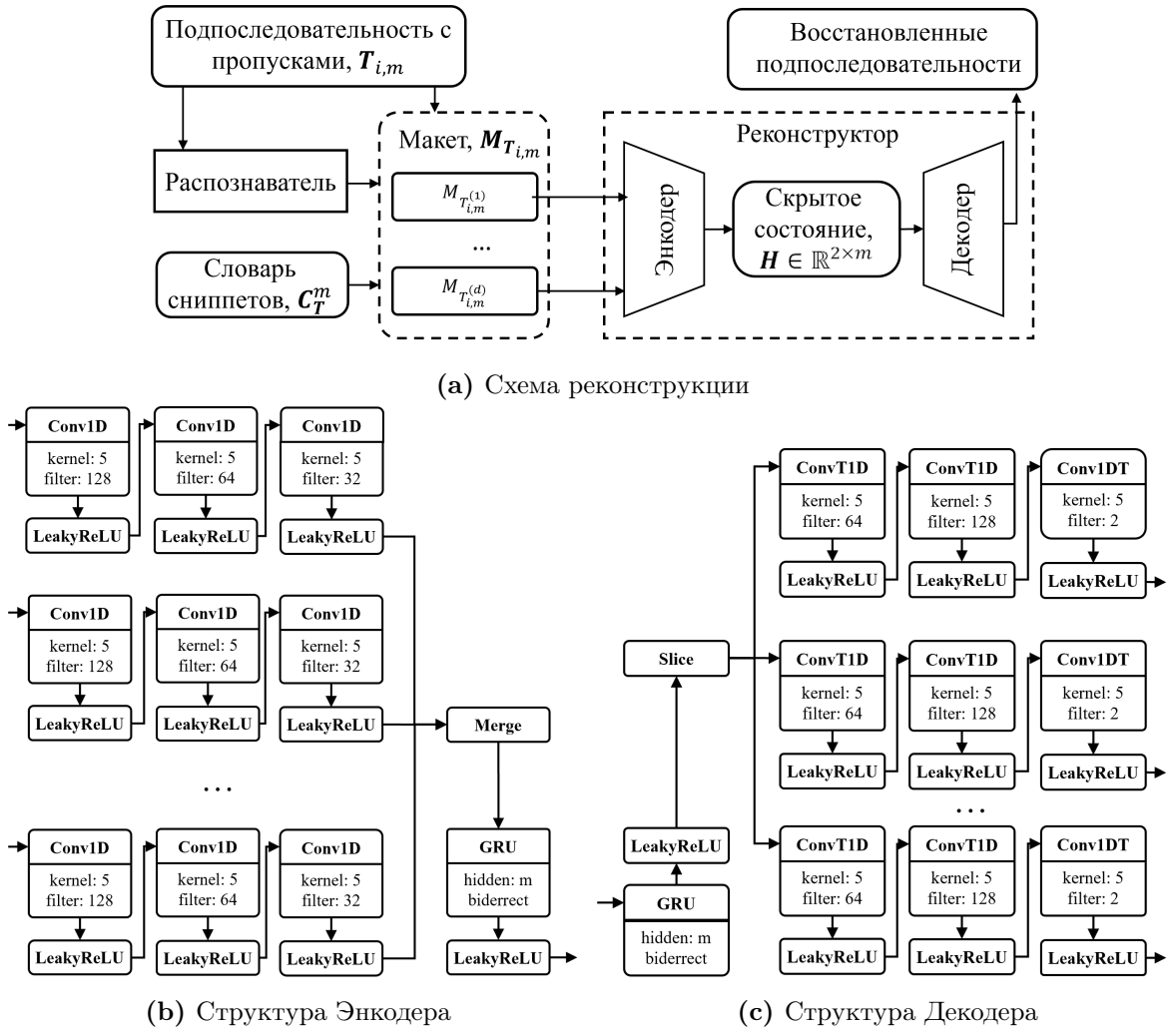
На рис. 2 представлена структура нейронной сети, реализующей Распознаватель. На вход сети поступает многомерная подпоследовательность ряда, пропущенные значения в которой заменены на ноль. На выходе сети для каждого измерения k входной подпоследовательности формируется *вектор активностей* $P_{T^{(k)}_{i,m}} \in \mathbb{R}^K$, состоящий из неотрицательных элементов, сумма которых равна 1. В данном векторе каждый элемент показывает вероятность, с которой одномерная подпоследовательность $T^{(k)}_{i,m}$, соответствующая измерению, принадлежит множеству ближайших соседей сниппета $C^m_{T^{(k)}}$, имеющего соответствующий номер.

Нейронная сеть состоит из следующих последовательно применяемых слоев: трех сверточных, одного рекуррентного и двух полносвязных. Три сверточных слоя, включающие 256, 128 и 64 карт признаков (feature maps) с размером ядра 5 соответственно, отвечают за извлечение признаков из данных входной подпоследовательности. После каждого сверточного слоя в целях уменьшения размерности данных и выделения наиболее важных признаков применяется операция подвыборки по максимальному значению (MaxPooling) с размером ядра 2. В качестве функции активации используется Линейный выпрямитель (ReLU, Rectified linear unit) [27]. Рекуррентный слой анализирует выделенные предыдущими слоями признаки с учетом временного контекста и реализуется с помощью управляемого рекуррентного блока (Gated Recurrent Units, GRU) [28] с длиной вектора скрытого состояния, равной 32. В качестве функции активации этого и всех последующих слоев применяется Линейный выпрямитель с «утечкой» (Leaky ReLU) [29].

Полносвязные слои содержат 32 и $d \cdot K$ нейронов соответственно. Слои, анализируя данные предыдущих слоев, формируют *матрицу активностей* $\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}} \in \mathbb{R}^{d \times K}$, в которой каждая строка представляет собой вектор активностей, рассмотренный выше. Матрица активностей имеет следующее формальное определение:

$$\begin{aligned} &\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}} = [\{P_{T^{(k)}_{i,\,m}}\}_{k=1}^{d}]^{\intercal}\colon\ P^{(k)}(j) = \Pr(T^{(k)}_{i,\,m} \in C^{m}_{T^{(k)}}), \\ &1 \leqslant i \leqslant n-m+1,\ 1 \leqslant k \leqslant d,\ 1 \leqslant j \leqslant K. \end{aligned} \tag{19}$$

3.4. Реконструктор



(a) Схема реконструкции

(b) Структура Энкодера

(c) Структура Декодера

Рис. 3. Структура Реконструктора

Рисунок 3 детализирует выполнение реконструкции многомерной подпоследовательности на этапе восстановления ряда. Схема реконструкции (см. рис. 3а) выглядит следующим образом.

Входная подпоследовательность передается на вход предварительно обученного Распознавателя, который выдает матрицу активностей. На основе указанной матрицы и словаря сниппетов, полученного в рамках предобработки, формируется вход Реконструктора, который, в свою очередь, выдает исходную подпоследовательность, где пропуски заменены на синтезированные значения. Данные, поступающие на вход Реконструктора, назовем *маке-*

том и определим как набор матриц, каждая из которых соответствует одному измерению входной многомерной подпоследовательности и состоит из двух строк. Первой строкой матрицы, входящей в макет, является одномерная подпоследовательность, а в качестве второй строки фигурирует соответствующий ей сниппет. В итоге формальное определение заготовки выглядит следующим образом:

$$\begin{aligned} \boldsymbol{M}_{\boldsymbol{T}_{i,\,m}} &= \{M_{T^{(k)}_{i,\,m}}\}_{k=1}^{d},\ M_{T^{(k)}_{i,\,m}} \in \mathbb{R}^{2\times m}, \\ M_{T^{(k)}_{i,\,m}}(1,\cdot) &= \hat{T}^{(k)}_{i,\,m},\ \boldsymbol{M}_{\hat{\boldsymbol{T}}_{i,\,m}} = \texttt{Zero}_{\beta\equiv 0}(\boldsymbol{T}_{i,\,m}), \\ M_{T^{(k)}_{i,\,m}}(2,\cdot) &= C^{m}_{T^{(k)}}\big(\arg\max_{1\leqslant j\leqslant K} P_{T^{(k)}_{i,\,m}}(j)\big),\ 1\leqslant k\leqslant d. \end{aligned} \tag{20}$$

Далее макет поступает на вход Энкодера для формирования скрытого состояния. Скрытое состояние макета представляет собой матрицу $H \in \mathbb{R}^{2\times m}$, в которой первая и вторая строки символизируют результаты обработки Энкодером входных данных, взятых в обычном и инвертированном порядке соответственно. Декодер, используя полученное скрытое состояние, формирует копию входной подпоследовательности, в которой пропущенные значения заменены синтезированными.

Обученный Распознаватель используется для обучения Реконструктора по следующим причинам. Полагается, что таким образом будет сформирована обучающая выборка Реконструктора, позволяющая адаптироваться модели к возможным ошибкам классификации, допускаемым Распознавателем, в процессе работы.

Структура Энкодера представлена на рис. 3b. Энкодер состоит d серий сверточных слоев и одного рекуррентного слоя. Каждая серия сверточных слоев принимает на вход соответствующую матрицу макета. Серия слоев состоит трех сверточных слоев, включающих 64, 32 и 1 карту признаков с размером ядра 5 соответственно. Задача каждой серии сверточных слоев сводится к выделению значимых признаков, важных для качественного сжатия, и связанных со схожестью входной одномерной подпоследовательности и соответствующего ей сниппета. Выходы серий сверточных слоев объединяются в матрицу размером $d \times m$, которая подается на вход рекуррентного слоя. Указанный слой состоит из двунаправленного GRU блока с размером скрытого состояния m. Рекуррентный слой анализирует извлеченные предыдущими слоями признаки, учитывая скрытые временные зависимости между измерениями подпоследовательности, и выдает скрытое состояние подпоследовательности.

Скрытое состояние поступает на вход Декодера (см. рис. 3c), структура которого представляет собой зеркальную копию Энкодера, в котором каждый сверточный слой заменен на соответствующий ему транспонированный сверточный слой. *Транспонированный сверточный слой (transposed convolutional layer)* [30] используется для увеличения пространственного разрешения входных данных. Транспонированная свертка представляет собой операцию, обратную свертке. Такой слой увеличивает размерность данных, вставляя нулевые значения между исходными входными значениями. Затем к получившимся данным применяется свертка, генерируя более крупные выходные данные. Выходы серий транспонированных сверточных слоев по всем измерениям объединяются в матрицу размером $d \times m$, представляющую собой восстановленную многомерную подпоследовательность.

4. Вычислительные эксперименты

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты на оборудовании Лаборатории суперкомпьютерного моделирования

ЮУрГУ [31]. В экспериментах исследовалась точность восстановления предложенного метода в различных предметных областях и сравнивалась с различными передовыми методами восстановления.

4.1. Описание экспериментов

Таблица 1. Наборы данных, используемые в экспериментах

№	Набор	Длина, $n \times 10^3$	Количество измерений, d	Предметная область
Группа А: Сезонность и цикл				
1.	BAFU [32]	50	10	Сброс воды в реках Швейцарии
2.	Climate [33]	5	10	Погода в различных локациях Северной Америки
3.	MAREL [34]	50	10	Характеристики морской воды в Ла-Манше
4.	MeteoSwiss [35]	10	10	Погода в городах Швейцарии
5.	Saaleaue [36]	23	14	Погода в городах Германии
Группа Б: Активности субъекта				
6.	Electricity [37]	5	9	Потребление электроэнергии в нескольких домашних хозяйствах
7.	Madrid [38]	25	10	Трафик автомобильных дорог в Мадриде
8.	Soccer [39]	100	10	Показания носимых датчиков футболистов во время матча
9.	WalkRun [11]	100	11	Показания датчиков смартфона во время прогулки/пробежки

Оценка предложенного метода и его сравнения с аналогами проводилась с использованием временных рядов, резюмированных в табл. 1. Временные ряды разделены на две группы в соответствии с особенностями данных: группа А включает в себя ряды 1–5, которые демонстрируют сезонность и цикл (циклические изменения уровня ряда с постоянным и переменным периодом соответственно); в группу Б входят ряды 6–9, для которых характерно отсутствие сезонности и циклических компонентов ввиду возможности случайного изменения активности субъекта. Целевой группой предложенного в данной работе метода SAETI являются временные ряды группы Б: ожидается, что SAETI покажет более высокую точность восстановления пропусков в тех данных, где могут быть выявлены устойчивые поведенческие шаблоны.

Формирование пропусков многомерного временного ряда осуществлялось в соответствии со сценарием MCAR (Missing Completely at Random, полностью случайное отсутствие), который был предложен в работе [7]. MCAR симулирует многократное отключение на непродолжительное время случайных источников данных. В итоге формируется множество относительно коротких пропусков (несколько подряд идущих точек) в случайных измерениях временного ряда. Для оценки точности восстановления в данной работе используется мера корня из среднеквадратичной ошибки RMSE (Root Mean Square Error) как одна из наиболее часто применяемых для этих целей метрик [40], определяемая следующим образом:

$$\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}, \tag{21}$$

где y_i — фактическое значение, $\hat{y}_i$ — восстановленное значение, h — количество восстановленных точек.

В экспериментах предложенный метод сравнивался с нейросетевыми методами восстановления NAOMI [21], BRITS [15], GP-VAE[22], M-RNN [16], SAITS и TRANSFORM [19] (исходные тексты реализации взяты из свободно доступных репозиториев, созданных авто-

рами данных разработок), а также передовыми аналитическими алгоритмами CDRec [13], DynaMMo [12], ORBITS [6], ROSL [41], GROUSE [42], SoftImpute [14], SVDImpute [43], SVT [44] и TeNMF [45] (исходные тексты реализации взяты из свободно доступного репозитория работы [7]).

Для проведения вычислительных экспериментов были установлены следующие параметры SAETI: размер входной подпоследовательности $m = 200$, количество активностей $K = 2$, доля подпоследовательностей ряда с пропусками $\alpha = 0.5$, доля обнуляемых элементов при подготовке обучающей выборки Реконструктора $\beta = 0.25$.

4.2. Анализ результатов

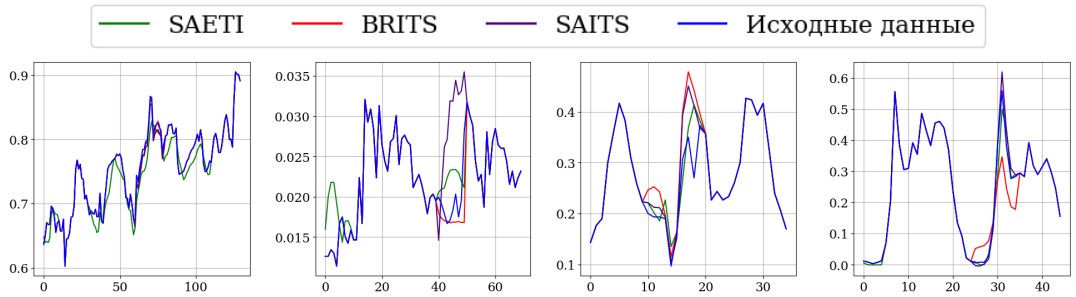
Таблица 2. Точность восстановления, RMSE·10^{-3}

Методы		Наборы данных											
		Группа А					Ср. ошибка	Группа Б				Ср. ошибка	Ср. ошибка
Тип	Название	BAFU	Climate	MAREL	MeteoSwiss	Saaleaue		Electricity	Madrid	Soccer	WalkRun		
Аналитические	CDRec	75(13)	156.2(12)	358(16)	78(9)	119(14)	157(15)	107(13)	103(14)	125(13)	177(15)	128(14)	144(14)
	DynaMMo	39(5)	142(8)	146(6)	73(5)	85.3(7)	97(7)	101(8)	71(6)	79(6)	127(6)	94(6)	96(6)
	GROUSE	206(16)	226(16)	214(13)	191(16)	155(16)	198(16)	134(15)	280(16)	163(15)	163(12)	185(16)	192(16)
	ORBITS	109(15)	166(14)	192(12)	85(11)	104(10)	131(13)	105(10)	90(11)	113.5(12)	174(14)	120.6(13)	126(13)
	ROSL	60(9)	179(15)	168(8)	101(13)	85(6)	119(10)	111(14)	99(13)	105(7)	135(8)	113(10)	116(10)
	SoftImpute	65(11)	152(10)	188(11)	80(10)	100(9)	117(9)	103(9)	76(8)	106(8)	156(10)	110(9)	114(9)
	SVDImpute	62(10)	156(11)	215(14)	77(7)	107(12)	123(12)	105.2(11)	80(10)	110(10)	161(11)	114(11)	119(11)
	SVT	74(12)	143(9)	180(10)	78(8)	75(5)	110(8)	95(7)	80(9)	108(9)	154(9)	109(8)	110(8)
	TeNMF	49(8)	163(13)	216(15)	77(6)	106(11)	122(11)	106(12)	91(12)	113(11)	168(13)	120(12)	121(12)
Нейросетевые	BRITS	17(4)	52(2)	**77(1)**	52(2)	**54(1)**	**50(1)**	**57(1)**	35(2)	20(5)	68(4)	45(2)	48(2)
	GP-VAE	46(7)	**15(1)**	172(9)	129(14)	92(8)	91(6)	74(4)	73(7)	142(14)	123(5)	103(7)	96.3(7)
	M-RNN	88(14)	138(7)	161(7)	169(15)	140(15)	139(14)	180(16)	108(15)	164(16)	227(16)	170(15)	153(15)
	NAOMI	40(6)	76(6)	128(5)	88(12)	107.1(13)	88(5)	75(5)	49(5)	9(2)	134(7)	67(5)	78(5)
	SAITS	15.2(2)	57.8(5)	79(2-3)	65(3-4)	57(2-3)	54.4(4)	73(3)	45(3-4)	14(3)	49(2-3)	45.1(3)	50(3)
	Transformer	**15(1)**	57(4)	79(2-3)	65(3-4)	57(2-3)	54(3)	81(6)	45(3-4)	20(4)	49(2-3)	49(4)	52(4)
	SAETI	16(3)	53(3)	88(4)	**52(1)**	58(4)	53(2)	60(2)	**31(1)**	**6(1)**	**42(1)**	**35(1)**	**45(1)**

Таблица 2 содержит сравнение точности методов восстановления на различных наборах данных. В таблице сравниваемые методы даны построчно и разделены на две группы: аналитические и нейросетевые конкуренты. По столбцам записаны две указанные выше группы наборов данных. В ячейках дается показатель точности и в скобках рейтинг данного результата среди всех конкурентов. Имеются столбцы среднего значения внутри каждой группы и по итогам экспериментов в целом. Лучшие результаты выделяются полужирным шрифтом.

Можно видеть, что в группе А (сезонность и цикл) метод BRITS показывает в среднем лучшую точность как среди аналитических, так и среди нейросетевых аналогов. Тем не менее, вторую позицию в этой группе занимает предложенный в данной работе метод SAETI, правда, несущественно при этом обгоняя методы SAITS и Transformer. Столь высокий результат SAETI в не целевой для себя группы данных, возможно, связан с тем, что в рамках сезона и (или) цикла временного ряда имели место колебания в данных, позволившие выявить существенно отличающиеся между собой поведенческие шаблоны, использование которых привело к увеличению точности восстановления. В целевой группе Б (активности)

предложенный метод уверенно обгоняет всех конкурентов, только в одном наборе данных показывая второй (после BRITS) результат.



(a) Saaleaue **(b)** MAREL **(c)** Electricity **(d)** Madrid

Рис. 4. Пример восстановленных рядов для различных наборов данных

Значимость поведенческих шаблонов проиллюстрирована на рис. 4, где изображены примеры фрагментов временных рядов, восстановленных с помощью SAETI и двух лучших конкурентов, BRITS и SAITS. При восстановлении рядов Saaleaue и MAREL (см. рис. 4a и 4b соответственно), в которых наблюдается возрастающий тренд и отсутствуют явные поведенческие шаблоны, SAETI уступает конкурентам по точности восстановления. Однако при обработке временных рядов Electricity и Madrid (см. рис. 4c и 4d соответственно), метод SAETI успешно использует шаблоны поведения людей в конкретный день недели, существенно обгоняя конкурентов.

Заключение

В данной статье затронута проблема разработки методов восстановления пропущенных значений в многомерных временных рядах, которая является актуальной в широком спектре предметных областей. Предложен новый метод восстановления, названный SAETI (Snippet-based Autoencoder for Time-series Imputation), который основан на совместном применении нейросетевых моделей-автоэнкодеров и аналитического поиска во временном ряде поведенческих шаблонов (сниппетов).

Метод SAETI предполагает фазы подготовки и восстановления данных. Во время первой фазы Препроцессор выполняет поиск сниппетов в каждом измерении входного временного ряда и формирует обучающие выборки нейросетевых моделей. Восстановление подпоследовательности, содержащей пропуски, заключается в последовательном применении к ней двух следующих нейросетевых моделей. Первая модель, Распознаватель, получает на вход подпоследовательность, в которой пропуски предварительно заменены на нули, и для каждого измерения определяет соответствующий сниппет. Распознаватель состоит из следующих последовательно применяемых слоев: трех сверточных, слоя GRU и двух полносвязных слоев. Вторая модель, Реконструктор, принимает на вход подпоследовательность и набор сниппетов, полученных Распознавателем, и заменяет пропуски временного ряда на правдоподобные синтетические значения.

Реконструктор реализован на основе автоэнкодеров, предполагающей два последовательно применяемых компонента: Энкодер и Декодер. Энкодер формирует скрытое состояние входной подпоследовательности и распознанных сниппетов. Декодер, принимая на вход скрытое состояние, производит восстановление исходной подпоследовательности. Энкодер включает в себя следующие слои: d серий сверточных слоев по три слоя в каждой серии и

общий слой GRU, где d — количество измерений. Декодер состоит из слоя GRU и d серий транспонированных сверточных слоев по три слоя в каждой серии.

Описаны вычислительные эксперименты, в которых предложенный метод сравнивался по точности восстановления пропусков с передовыми аналитическими алгоритмами и нейросетевыми моделями на реальных временных рядах из различных предметных областей. Результаты экспериментов показывают, что SAETI в среднем опережает остальных конкурентов и показывает лучшие результаты в случае, когда восстанавливаются данные, отражающие активность некоего субъекта.

В будущих исследованиях планируется изучить влияние способов разбиения временных рядов, содержащих пропущенные значения, при формировании обучающих выборок моделей, на точность восстановления.

Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).

Литература

1. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review // J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/S40537-019-0268-2.
2. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems // J. Aerosp. Inf. Syst. 2024. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.I011320.
3. Zhou Z., Tang W., Li M., *et al.* A Novel Hybrid Intelligent SOPDEL Model with Comprehensive Data Preprocessing for Long-Time-Series Climate Prediction // Remote. Sens. 2023. Vol. 15, no. 7. P. 1951. DOI: 10.3390/RS15071951.
4. Majumdar S., Laha A.K. Clustering and classification of time series using topological data analysis with applications to finance // Expert Syst. Appl. 2020. Vol. 162. P. 113868. DOI: 10.1016/J.ESWA.2020.113868.
5. Yen N.Y., Chang J., Liao J., Yong Y. Analysis of interpolation algorithms for the missing values in IoT time series: a case of air quality in Taiwan // J. Supercomput. 2020. Vol. 76, no. 8. P. 6475–6500. DOI: 10.1007/S11227-019-02991-7.
6. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams // Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.
7. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series // Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.
8. Fang C., Wang C. Time Series Data Imputation: A Survey on Deep Learning Approaches // CoRR. 2020. Vol. abs/2011.11347. arXiv: 2011.11347. URL: https://arxiv.org/abs/2011.11347.
9. Wang J., Du W., Cao W., *et al.* Deep Learning for Multivariate Time Series Imputation: A Survey // CoRR. 2024. Vol. abs/2402.04059. DOI: 10.48550/ARXIV.2402.04059. arXiv: 2402.04059.

10. Imani S., Madrid F., Ding W., *et al.* Introducing time series snippets: A new primitive for summarizing long time series // Data Min. Knowl. Discov. 2020. Vol. 34, no. 6. P. 1713–1743. DOI: 10.1007/s10618-020-00702-y.

11. Цымблер М.Л., Юртин А.А. Восстановление пропущенных значений временного ряда на основе совместного применения аналитических алгоритмов и нейронных сетей // Вычислительные методы и программирование. 2023. Т. 24, № 3. С. 243—259. DOI: 10.26089/NumMet.v24r318.

12. Li L., McCann J., Pollard N.S., Faloutsos C. DynaMMo: mining and summarization of coevolving sequences with missing values // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 507–516. DOI: 10.1145/1557019.1557078.

13. Khayati M., Cudré-Mauroux P., Böhlen M.H. Scalable recovery of missing blocks in time series with high and low cross-correlations // Knowl. Inf. Syst. 2020. Vol. 62, no. 6. P. 2257–2280. DOI: 10.1007/S10115-019-01421-7.

14. Mazumder R., Hastie T., Tibshirani R. Spectral Regularization Algorithms for Learning Large Incomplete Matrices // J. Mach. Learn. Res. 2010. Vol. 11. P. 2287–2322. DOI: 10.5555/1756006.1859931.

15. Cao W., Wang D., Li J., *et al.* BRITS: Bidirectional Recurrent Imputation for Time Series // Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada / ed. by S. Bengio, H.M. Wallach, H. Larochelle, *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.

16. Yoon J., Zame W.R., Schaar M. van der Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks // IEEE Trans. Biomed. Eng. 2019. Vol. 66, no. 5. P. 1477–1490. DOI: 10.1109/TBME.2018.2874712.

17. Luo Y., Zhang Y., Cai X., Yuan X. E^2GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation // Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019 / ed. by S. Kraus. ijcai.org, 2019. P. 3094–3100. DOI: 10.24963/IJCAI.2019/429.

18. Wu Z., Ma C., Shi X., *et al.* BRNN-GAN: Generative Adversarial Networks with Bidirectional Recurrent Neural Networks for Multivariate Time Series Imputation // 27th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2021, Beijing, China, December 14-16, 2021. IEEE, 2021. P. 217–224. DOI: 10.1109/ICPADS53394.2021.00033.

19. Du W., Côté D., Liu Y. SAITS: Self-attention-based imputation for time series // Expert Syst. Appl. 2023. Vol. 219. P. 119619. DOI: 10.1016/J.ESWA.2023.119619.

20. Oh E., Kim T., Ji Y., Khyalia S. STING: Self-attention based Time-series Imputation Networks using GAN // IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021 / ed. by J. Bailey, P. Miettinen, Y.S. Koh, *et al.* IEEE, 2021. P. 1264–1269. DOI: 10.1109/ICDM51629.2021.00155.

21. Liu Y., Yu R., Zheng S., *et al.* NAOMI: Non-Autoregressive Multiresolution Sequence Imputation // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada / ed. by H.M. Wallach, H. Larochelle, A. Beygelzimer, *et al.* 2019. P. 11236–11246. URL: https://proceedings.neurips.cc/paper/2019/hash/50c1f44e426560f3f2cdcb3e19e39903-Abstract.html.

22. Fortuin V., Baranchuk D., Rätsch G., Mandt S. GP-VAE: Deep Probabilistic Time Series Imputation // The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Vol. 108 / ed. by S. Chiappa, R. Calandra. PMLR, 2020. P. 1651–1661. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v108/fortuin20a.html.

23. Kingma D.P., Welling M. Auto-Encoding Variational Bayes // CoRR. 2013. Vol. abs/1312.6114. URL: https://api.semanticscholar.org/CorpusID:216078090.

24. Roberts S.J., Osborne M.A., Ebden M., *et al.* Gaussian processes for time-series modelling // Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2013. Vol. 371. URL: https://api.semanticscholar.org/CorpusID:556194.

25. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

26. Benschoten A.V., Ouyang A., Bischoff F., Marrs T. MPA: a novel cross-language API for time series analysis // Journal of Open Source Software. 2020. Vol. 5, no. 49. P. 2179. DOI: 10.21105/joss.02179.

27. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/S0218488598000094.

28. Chung J., Gülçehre Ç., Cho K., Bengio Y. Gated Feedback Recurrent Neural Networks // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Vol. 37 / ed. by F.R. Bach, D.M. Blei. JMLR.org, 2015. P. 2067–2075. JMLR Workshop and Conference Proceedings. URL: http://proceedings.mlr.press/v37/chung15.html.

29. Guo Y., Li S., Lerman G. The effect of Leaky ReLUs on the training and generalization of overparameterized networks // International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain. Vol. 238 / ed. by S. Dasgupta, S. Mandt, Y. Li. PMLR, 2024. P. 4393–4401. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v238/guo24c.html.

30. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning // CoRR. 2016. Vol. abs/1603.07285. arXiv: 1603.07285. URL: http://arxiv.org/abs/1603.07285.

31. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университет // Вычислительные методы и программирование. 2022. Т. 11, № 1. С. 15—30. DOI: 10.14529/cmse220102.

32. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. Accessed: 2023-09-03. https://www.hydrodaten.admin.ch/.

33. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

34. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. Accessed: 2023-09-03 DOI: 10.17882/39754.

35. MeteoSwiss: Federal Office of Meteorology and Climatology. 2023. Accessed: 2023-09-03. https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html.

36. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. Accessed: 2023-09-03. https://www.bgc-jena.mpg.de/wetter/weather_data.html.

37. Trindade A. Electricity Load Diagrams 2011–2014. 2015. DOI: 10.24432/C58C86. UCI Machine Learning Repository.

38. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

39. Mutschler C., Ziekow H., Jerzak Z. The DEBS 2013 grand challenge // The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA, June 29 - July 03, 2013 / ed. by S. Chakravarthy, S.D. Urban, P.R. Pietzuch, E.A. Rundensteiner. ACM, 2013. P. 289–294. DOI: 10.1145/2488222.2488283.

40. Minor B.D., Doppa J.R., Cook D.J. Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications // IEEE Trans. Knowl. Data Eng. 2017. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/TKDE.2017.2750669.

41. Shu X., Porikli F., Ahuja N. Robust Orthonormal Subspace Learning: Efficient Recovery of Corrupted Low-Rank Matrices // 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014. IEEE Computer Society, 2014. P. 3874–3881. DOI: 10.1109/CVPR.2014.495.

42. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case // Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

43. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays // Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/BIOINFORMATICS/17.6.520.

44. Cai J., Candès E.J., Shen Z. A Singular Value Thresholding Algorithm for Matrix Completion // SIAM J. Optim. 2010. Vol. 20, no. 4. P. 1956–1982. DOI: 10.1137/080738970.

45. Mei J., Castro Y. de, Goude Y., Hébrail G. Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, 2017. P. 2382–2390. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v70/mei17a.html.

Юртин Алексей Артемьевич, программист, Лаборатория больших данных и машинного обучения, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240203

IMPUTATION OF MULTIVARIATE TIME SERIES BASED ON THE BEHAVIORAL PATTERNS AND AUTOENCODERS

© 2024 A.A. Yurtin

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: iurtinaa@susu.ru

Received: 01.05.2024

Currently, in a wide range of subject domains, the problem of imputation missing points or blocks of time series is topical. In the article, we present SAETI (Snippet-based Autoencoder for Time-series Imputation), a novel method for imputation of missing values in multidimensional time series that is based on the combined use of autoencoders and a time series of behavioral patterns (snippets). The imputation of a multidimensional subsequence is performed using the following two neural network models: The Recognizer, which receives a subsequence as input, where the gaps are pre-replaced with zeros, and determines the corresponding snippet for each dimension; and the Reconstructor, which takes as input a subsequence and a set of snippets received from the Recognizer, and replaces the missing elements with plausible synthetic values. The Reconstructor is implemented as a combination of the following two models: An Encoder that forms a hidden state for a set of input sequences and recognized snippets; and a Decoder that receives a hidden state as input, which imputes the original subsequence. In the article, we present a detailed description of the above models. The results of experiments over time series from real-world subject domains showed that SAETI is on average ahead of state-of-the-art analogs in terms of accuracy and shows better results when input time series reflect the activity of a certain subject.

Keywords: time series, imputation of missing values, autoencoders, behavioral patterns (snippets) of time series, neural networks.

FOR CITATION

Yurtin A.A. Imputation of Multivariate Time Series Based on the Behavioral Patterns and Autoencoders. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 39–55. (in Russian) DOI: 10.14529/cmse240203.

УДК 004.932.72'1, 004.021, 519.254 DOI: 10.14529/cmse240204

ГИБРИДНЫЙ АЛГОРИТМ РАСПОЗНАВАНИЯ СТРОЕНИЙ НА СПУТНИКОВЫХ СНИМКАХ НА ОСНОВЕ МЕТОДА ЖУКА И АЛГОРИТМА ИСКЛЮЧЕНИЯ ОБЛАСТЕЙ

© 2024 И.В. Баранова, С.В. Гилин

Сибирский федеральный университет
(660041 Красноярск, пр. Свободный, д. 79)
E-mail: ibaranova@sfu-kras.ru, gilin.stepan@mail.ru

Поступила в редакцию: 10.01.2024

В статье предлагается новый метод распознавания строений на спутниковых снимках. Представленный метод является гибридным, он основан на алгоритме исключения областей и методе жука. Алгоритм исключения областей представляет собой хорошо известный и эффективный способ сегментации изображения на регионы схожих пикселей по различным признакам: цвет, текстура, яркость, форма и др. Метод жука — классический метод контурного анализа, выполняющий последовательное вычерчивание границы между объектом и фоном. В рамках работы предлагаемого алгоритма сначала метод исключения областей выделяет потенциальные области, в которых могут находиться строения и устраняет нежелательные элементы на изображении (растительность, водные поверхности и дороги), которые могут быть ложно распознаны как строения. Далее модифицированный метод жука определяет местоположение и контуры строений. На финальном этапе среди обнаруженных объектов выявляются искусственно созданные объекты, у которых имеется объем. Для реализации проверки объектов на искусственное происхождение и объемность разработаны собственные методы. Представленный алгоритм распознавания показывает хорошую точность распознавания и не требует обучающей выборки. В статье описывается программная реализация предлагаемого метода. Демонстрируются результаты вычислительных экспериментов по оцениванию эффективности метода и сравнению с тремя известными алгоритмами распознавания.

Ключевые слова: распознавание строений, спутниковые снимки, гибридный метод, метод жука, метод исключения областей, текстурные характеристики, границы зданий, потенциальные области, классификация.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Баранова И.В., Гилин С.В. Гибридный алгоритм распознавания строений на спутниковых снимках на основе метода жука и алгоритма исключения областей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 56–76. DOI: 10.14529/cmse240204.

Введение

В настоящее время задача обнаружения на спутниковых снимках зданий и различных строений является одной из актуальных и практически важных задач в области распознавания образов, компьютерного зрения, искусственного интеллекта и анализа данных. Поставленная задача распознавания заключается в обнаружении объектов заданных классов на изображении. Распознавание указанных объектов на аэрокосмических снимках может применяться для решения широкого круга практических задач в различных областях деятельности: ориентации на местности, составлении карт (топологического дешифрирования снимков), обследовании и строительном контроле зданий и сооружений, нахождении оптимального маршрута движения транспорта, мониторинга инженерных сетей в построенных комплексах, определении законности строений и других. Так, например, в работе [1] нами

решается практический пример подобной задачи — задача распознавания строений, находящихся в водоохранных зонах.

Решение задачи распознавания объектов на изображениях выполняется в несколько этапов: предварительная обработка исходных изображений; сегментация изображения (группировка областей с одинаковыми визуальными характеристиками); обнаружение объектов и извлечение векторов признаков для обнаруженных объектов; классификация объектов на основании выделенных признаков. Этапы обнаружения и классификации объектов чаще всего формулируются как структурные и оптимизационные задачи на многослойных объектах, что, несомненно, увеличивает сложность решаемой задачи распознавания. Поэтому разработка новых и усовершенствование существующих автоматических алгоритмов распознавания аэрокосмических снимков высокого разрешения, способных находить решение задачи за реальное время, является одним из актуальных и востребованных направлений в широком спектре предметных областей, связанных с анализом визуальных данных.

Целью данной работы является разработка нового алгоритма распознавания строений на спутниковых снимках. Для достижения цели решаются следующие задачи:

1. разработка и обоснование собственного алгоритма распознавания изображений как синтеза метода жука и алгоритма исключения областей;
2. создание программного обеспечения на языке Python, реализующего работу предложенного метода распознавания;
3. проведение вычислительных экспериментов по оцениванию характеристик разработанного алгоритма, а также его сравнению с тремя наиболее распространенными методами распознавания по скорости работы и точности распознавания.

Статья организована следующим образом. Она состоит из введения, пяти разделов и заключения. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 описана общая постановка задачи. В разделе 3 рассмотрена предобработка изображений и существующие методы решения задачи. В разделе 4 описан собственный алгоритм распознавания строений на спутниковых снимках. В разделе 5 описаны результаты работы предложенного алгоритма, описаны вычислительные эксперименты и результаты сравнения предложенного алгоритма с некоторыми из существующих алгоритмов. Заключение содержит краткое изложение полученных результатов и выводы об их применении.

1. Обзор работ по теме исследования

На данный момент существует ряд методов, успешно решающих задачу распознавания объектов на спутниковых снимках. Одним из широко используемых подходов к автоматическому распознаванию снимков является группа методов, основанных на разрезах в графах. Ярким представителем этой группы является алгоритм сегментации спутниковых изображений на основе суперпикселей и разрезов на графах, представленный в работе [2]. Несомненным преимуществом алгоритмов графового подхода является сокращение времени на классификацию и детекцию объектов, но из-за представления каждого пикселя изображения в виде вершины графа увеличивается вычислительная сложность задачи. Еще одним известным подходом к решению поставленной задачи служат методы исключения областей [3, 12], основывающиеся на сегментации изображения с разделением на регионы, объединяющие схожие пиксели по различным признакам, таким как цвет, текстура или форма. Также в этих методах предполагается, что объекты и фон имеют разные интен-

сивности пикселей или цветовые характеристики, что позволяет легко выделить объекты, установив пороговое значение для пикселей или цветов.

В последние годы одним из самых популярных подходов к распознаванию спутниковых снимков является нейросетевой подход. Нейронные сети различных архитектур были успешно применены для обработки многопиксельных спутниковых изображений высокого разрешения и позволили найти эффективное решение задачи сегментации объектов. В том числе ряд современных нейросетевых архитектур использовался и непосредственно для решения задач обнаружения зданий на снимках — например, архитектуры UNet, SegNet и Mask R-CNN. Так, в статье [4] применяется архитектура SegNet. Довольно часто в рамках данного подхода использование нейронных сетей сочетается с применением методов машинного обучения, преимущественно — методов глубокого обучения. Например, в работе [5] используется ассиметричная CNN. На данный момент одной из лучших нейросетей обнаружения строений является архитектура UANet + PVT-V2-B2 [6], поэтому в данной работе будет выполняться сравнение предлагаемого нами алгоритма распознавания именно с указанной нейросетью. В целом, можно отметить, что нейросетевые методы успешно справляются с распознаванием зданий и строений на аэроснимках, но для высокой точности распознавания требуется иметь достаточно объемную обучающую выборку, что в некоторых практических задачах является существенной проблемой.

Еще одним активно используемым подходом к решению задачи распознавания объектов на спутниковых снимках являются методы с выделением контуров объектов: алгоритмы активных контуров (например, метод затравочных точек или метод на основе энергии уровней) или алгоритмы водораздела. Модели, основанные на активных контурах, показывают высокую точность распознавания вне зависимости от качества изображения [7]. Данные методы не требуют обучающей выборки для своей работы. Ряд авторов предлагает методы, комбинирующие в себе часть из перечисленных подходов. К примеру, в работе [8] применялась структура из замкнутых контуров с дальнейшей постобработкой. Как правило, подобное комбинирование позволяет избавиться от недостатков, присущих определенному подходу, и получить более точные результаты распознавания. Например, часто используется пороговая сегментация для грубой предварительной сегментации, затем для уточнения результатов применяются более сложные методы, такие как нейросети или графовые алгоритмы, как представлено в работе [9].

В данной работе предлагается собственный алгоритм распознавания строений на спутниковых снимках, основанный на алгоритме исключения областей, с использованием собственной модификации метода обнаружения контуров — метода жука. Предлагаемый алгоритм распознавания будет базироваться на версии алгоритма исключения областей, предложенного в работе [3].

2. Постановка задачи

Как уже было сказано выше, в данной работе решается задача распознавания строений на изображении. Приведем необходимые определения, позволяющие уточнить условия поставленной задачи.

2.1. Необходимые уточнения для формулировки задачи

В Федеральном законе РФ от 30 декабря 2009 года N 384-ФЗ «Технический регламент о безопасности зданий и сооружений» дается следующее определение: сооружение — на-

земная или подземная система, выполненная методом строительства и предназначенная для проживания и/или хозяйственной деятельности людей. Для сооружений также можно использовать термины «искусственная структура» или «искусственный объект». В том же законе под строением понимается любое объемное наземное строительное сооружение, технологически имеющее неразрывную связь с грунтом, а под зданием понимается неподвижное наземное строение, имеющее внутренний полезный объем, а также имеющее сети и системы инженерно-технического обеспечения. Таким образом, в рамках задачи, которая решается в данной работе, будет выполняться распознавание наземных строений с внутренним полезным объемом. К числу таких строений будем относить здания (жилые и нежилые), сараи, гаражи, бани и др. Суть решаемой задачи заключается в следующем: алгоритму на вход поступает спутниковый снимок, после чего он должен обнаружить на данном изображении все строения, выделив их контуры. После детекции некоторого объекта на изображении выделяются его признаки (цвет пикселей, текстура, углы объекта, форма контура), на основании которых затем выполняется классификация обнаруженного объекта — т.е. отнесение его к категории искусственно созданных объектов. Данная классификация возможна благодаря свойству, которым обладают все искусственные объекты: они либо имеют малую постоянную кривизну (объекты круглой формы), либо несколько точек высокой кривизны (объекты прямоугольной формы) или вообще не имеют кривизны. Также стоит отметить, что мы выполняем распознавание строений, которые в отличие от линейных и плоскостных сооружений обязательно имеют внутренний объем, что позволяет их явным образом отличать от большинства инженерно-технических сооружений. Примерами линейных сооружений являются линии электропередач, мосты, телевизионные башни, мачты и тому подобное. К плоскостным, например, относятся оборудованные стоянки автомашин. Признаком того, что строение имеет объем, является наличие у строения на спутниковом снимке тени, которая в зависимости от времени суток меняет свое расположение и форму. Таким образом, для отнесения обнаруженного искусственного объекта к строениям необходимо наличие у него тени.

2.2. Математическая постановка задачи распознавания образов

Пусть $X = \{x_1, x_2, \ldots, x_n\}$ — множество объектов распознавания, $x : x \in X$ — объект распознавания. Каждый объект $x \in X$ описывается вектором признаков, т.е. задано F — пространство признаков, функция $f(x) : X \to F$ ставит в соответствие каждому объекту $x \in X$ точку $f(x)$. Таким образом, вектор $f(x)$ представляет собой образ объекта $x \in X$.

Пусть в пространстве признаков F определено множество классов $C = \{C_1, \ldots, C_m\}$, $C_i \in F,\ i = 1, \ldots, m,\ C_i \cap C_j = \oslash$ при $i \neq j$. Функция классификации $g(x) : X \to C$ — неизвестная индикаторная функция, разбивающая пространство образов F на m непересекающихся классов C_i. Решающее правило $\hat{g}(f) : F \to C$ — функция, которая является оценкой для $g(x)$ на основании образа объекта, то есть $\hat{g}(f) = \hat{g}(f(x))$. Пусть задано множество $\widehat{X} = \{x_j\}$, $j = 1, \ldots, k$, $k < n$, $\widehat{X} \subset X$, для которого задана информация о функциях $f(x)$ и $g(x)$, т.е. $f_j = f(x_j)$ и $g_j = f(g_j)$, для $x_j \in \widehat{X}$, $j = 1, \ldots, k$. Множество (g_j, f_j) называется множеством прецедентов.

Требуется построить такое решающее правило (*алгоритм классификации*) $\hat{g}(f)$, чтобы $Q(\hat{g}, C) = \min_f Q(g, C)$, где $Q(g, C)$ — выбранный критерий качества классификации. В данной работе в качестве критерия используются четыре метрики (точность, полнота, усредненный индекс Жаккара и мера F1), представляющие собой функции от количества

ошибок первого и второго рода [10]. Ошибка первого рода — ситуация, когда происходит ложное срабатывание алгоритма классификации на объекте, т.е. обнаруженный объект класса фактически отсутствует на изображении. В статистическом контексте это можно интерпретировать как отклонение верной гипотезы об отсутствии объекта искомого класса на изображении. Ошибка второго рода заключается в упущении объекта, который действительно присутствует на изображении, т.е. алгоритм не обнаруживает его. Определения вышеуказанных метрик приводятся в разделе 5.

2.3. Особенности распознавания объектов на изображениях

Итак, в качестве объектов распознавания $x \in X$ выступают строения. Как уже было сказано выше, распознавание строений выполняется на спутниковых снимках. В рамках данной работы будут использоваться растровые изображения. Пусть имеется растровое изображение D размером $w \times h$, представленное двумерной матрицей пикселей, принадлежащих цветовому пространству RGB. Т.е. изображение задается в следующем виде: $D = \{p_{i,j}, i = 1, \ldots, h, j = 1, \ldots, w, \}$, где $p_{i,j}$ — пиксель, (i, j) — координаты пикселя на изображении, w, h — ширина и высота изображения, $p_{i,j} = \{R(p_{i,j}) \in [0, 255], G(p_{i,j}) \in [0, 255], B(p_{i,j}) \in [0, 255]\}$. Изображение содержит множество объектов распознавания $X = \{x_1, x_2, \ldots, x_n\}$: $X \subseteq D$. Формально каждый объект распознавания $x_k \in X$ можно представить в виде множества пикселей: $x_k = \{p_{ij}, i = 1, \ldots, h_k, j = 1, \ldots, w_k\}$, где w_k, h_k — ширина и высота объекта.

2.4. Методы предварительной обработки и фильтрации изображения

Перед выполнением распознавания выполняется предварительная обработка изображения для устранения шумов и дефектов изображения. Шумы на изображении появляются в самых разнообразных формах, поэтому существует множество методов шумоподавления. В данной работе для устранения шума изображения используется метод скалирования изображения, который опирается на значения пикселей в исходном изображении и вычисляет новые значения для пикселей в увеличенном и уменьшенном изображении на основе окружающих пикселей.

Для выявления объектов с однотипным внешним видом используется сегментация изображения, т.е. группировка областей с одинаковыми визуальными характеристиками. В работе [11] показано, что одного набора параметров недостаточно для корректного сегментирования каждого здания на снимке. Если здания имеют сложную форму, иногда бывает невозможно сегментировать даже одно здание по одному набору параметров. Для упрощения определения признаков выполним фильтрацию изображения. Для фильтрации изображения выбран метод сдвига среднего значения. Сдвиг среднего значения — непараметрический метод анализа пространства признаков для определения максимума плотности вероятности [12]. Он используется для уменьшения шума, сглаживания текстур, улучшения контрастности. Данные свойства способствуют более эффективному распознаванию объектов на изображениях. Для программной реализации данного метода в работе используется функция pyrMeanShiftFiltering() открытой библиотеки компьютерного зрения и обработки изображений OpenCV. Результаты применения предварительной обработки и выбранного метода фильтрации к исходному изображению показаны на рис. 1. Слева на рисунке представлено исходное спутниковое изображение, по центру — предобработанное и сглаженное изображение.

а) Исходное изображение б) Сглаженное изображение в) Преобразование к схеме HSV

Рис. 1. Предварительная обработка, сглаживание и фильтрация изображения

В качестве следующего шага фильтрации изображения выполним преобразование цветовых компонент изображения к цветовой модели HSV. В схеме HSV цвет представляется точкой в трехмерном пространстве (H, S, V), где H (Hue) — это цветовой тон, S (Saturation) — насыщенность, а V (Value) — яркость цвета. Указанная цветовая модель благодаря своей специфике позволяет более корректно определить порог необходимого значения для сегментации изображения, которое будет выполняться на следующем этапе задачи распознавания с помощью предлагаемого нами алгоритма распознавания. На рис. 1 справа продемонстрирован результат преобразования изображения к схеме HSV, причем выполняется преобразование изображения, уже прошедшего предобработку и сглаживание сдвигом среднего значения (т.е. выполняется преобразование изображения, приведенного на рис. 1б).

3. Модифицированный алгоритм синтеза метода «жука» и метода исключения областей

В работе предлагается новый метод автоматического обнаружения строений на спутниковых снимках, который является модифицированным гибридом алгоритма исключения областей и метода жука. Как уже было сказано ранее, в рамках поставленной задачи нам необходимо найти на изображении строения, которые относятся к искусственным объектам. Кроме того, алгоритм должен учесть требование из определения строения (из раздела 2.1), т.е. то, что строение обязательно имеет внутренний объем, что позволяет явным образом отличать его от большинства инженерно-технических сооружений. Исходя из всего вышесказанного, последовательность этапов работы предложенного алгоритма выглядит следующим образом:

1. Сегментация изображения, т.е. группировка областей с одинаковыми визуальными характеристиками и выявление потенциальных областей, в которых могут находиться искомые объекты (строения);
2. Обнаружение объектов и выделение их контуров;
3. Проверка контуров найденных объектов на кривизну для выявления искусственно созданных объектов;
4. Проверка найденных искусственно созданных объектов на наличие у них объема.

На первом этапе работы разработанного алгоритма распознавания для выполнения сегментации изображения используется метод исключения областей. На втором этапе для обнаружения объектов и выделения их контуров используется предложенный нами модифицированный алгоритм жука. Для выявления искусственно созданных объектов нами разработан алгоритм проверки контуров объектов на кривизну, а для выявления среди обна-

руженных искусственных объектов строений (в т.ч. зданий) — алгоритм проверки объекта на то, что у него есть объем.

3.1. Метод исключения областей

Метод исключения областей представляет собой хорошо известный и эффективный подход к распознаванию объектов на изображении. Он позволяет выявить потенциальные области, где могут находиться искомые объекты, основываясь на изменениях интенсивности пикселей и текстурных переходах, связанных с контурами объектов. Этот алгоритм будет применяться для удаления нежелательных элементов на изображении, таких как деревья, водные поверхности и дороги, которые могут быть ложно распознаны как здания. В нашей работе предлагается модификация метода исключения областей, базирующаяся на методе, приведенном в работе [3]. Оригинальный метод выполняет бинаризацию исходного полутонового изображения путем сравнения яркости каждого пикселя с заданным пороговым значением. Пиксели со значением яркости ниже порога окрашиваются черным цветом, в противном случае — белым. Отличия предлагаемой нами модификации метода заключаются в следующем.

1. На вход алгоритму поступает цветное изображение в схеме HSV.
2. Для каждого цветового тона (красного, зеленого и синего) устанавливается интервал значений яркости и насыщенности, при помощи которого можно выделить участки с необходимыми спектральными характеристиками.
3. Для каждого цвета на основе найденных интервалов задаются пороговые значения для яркости и насыщенности.
4. Для каждого пикселя изображения определяется его цветовой тон (путем проверки попадания в соответствующие интервалы значений), затем выполняется сравнение его яркости с пороговым значением яркости данного цвета и его насыщенности с пороговым значением насыщенности данного цвета. Если значение яркости ниже соответствующего порога, а насыщенности — ниже, то пиксель окрашивается белым цветом, в противном случае — черным. Это условие объясняется тем, что искусственные объекты имеют большую цветовую насыщенность, чем естественные объекты.

Введем обозначения, необходимые для описания алгоритма. Для корректной работы алгоритма необходимо задать интервалы значений каждого цветового тона: $HR_{\min}$ — начало диапазона красного цвета, $HR_{\max}$ — конец диапазона красного цвета, $HG_{\min}$ и $HG_{\max}$ — начало и конец диапазона зеленого цвета, $HB_{\min}$ и $HB_{\max}$ — начало и конец диапазона синего цвета. Указанные значения задаются экспертно на основе знаний о характеристиках изображений: $HR_{\min} = 300$, $HR_{\max} = 360$, $HG_{\min} = 90$, $HG_{\max} = 164$, $HB_{\min} = 165$, $HB_{\max} = 255$. В переменных $SR_{\min}$ и $SR_{\max}$ хранятся начало и конец диапазона насыщенности красного цвета, в $SG_{\min}$ и $SG_{\max}$ — начало и конец диапазона насыщенности зеленого цвета, в $SB_{\min}$ и $SB_{\max}$ — начало и конец диапазона насыщенности синего цвета. Переменные $VR_{\min}$, $VR_{\max}$, $VG_{\min}$, $VG_{\max}$, $VB_{\min}$, $VB_{\max}$ задают границы диапазонов яркости для каждого из перечисленных цветов. Также для работы алгоритма используются пороговые переменные: TSR — пороговое значение насыщенности красных оттенков, TVR — пороговое значение яркости красных оттенков. Схожим образом определяются переменные TSG и TVG, TSB и TVB. Обозначим получение порогового значения насыщенности как $TS(f) = 3/4 * (f_{\max} - f_{\min})$, а получение порогового значения яркости как $TV(f) = 1/3 * (f_{\max} - f_{\min})$. Изображение в схеме HSV задается следующим образом: $D_{\text{HSV}} = \{d_{i,j},\ i = 1, \ldots, h,\ j = 1, \ldots, w, \}$, где $d_{i,j}$ — пик-

сель, (i, j) — координаты пикселя на изображении, w, h — ширина и высота изображения, $d_{i,j} = \{H(p_{i,j}) \in [0, 360]\,,\, S(p_{i,j}) \in [0, 100]\,,\, V(p_{i,j}) \in [0, 100]\}$. $D_{\text{Binary}} = \{p_{ij},\, i = 1, \ldots, h,\, j = 1, \ldots, w\}$, $p_{ij} = 0$ или 1 — получаемое бинарное изображение.

Опишем этапы работы предлагаемой модификации метода исключения областей Exclusion:

Вход алгоритма: D_{HSV} — цветное изображение в схеме HSV.

Выход алгоритма: D_{Binary} — бинарное изображение.

Шаг 1. $i := 1$; $j := 1$; $SR_{\min} := 100$; $SG_{\min} := 100$; $SB_{\min} := 100$; $SR_{\max} := 0$; $SG_{\min} := 0$; $SB_{\min} := 0$; $D_{\text{Binary}} := \{0\}$.

Шаг 2. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $S(d_{ij}) < SR_{\min}$, то $SR_{\min} := S(d_{ij})$.

Шаг 3. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $S(d_{ij}) > SR_{\max}$, то $SR_{\max} := S(d_{ij})$; перейти на шаг 14.

Шаг 4. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $V(d_{ij}) < VR_{\min}$, то $VR_{\min} := V(d_{ij})$.

Шаг 5. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $V(d_{ij}) > VR_{\max}$, то $VR_{\max} := V(d_{ij})$; перейти на шаг 14.

Шаг 6. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $H(d_{ij}) < SG_{\min}$, то $SG_{\min} := H(d_{ij})$.

Шаг 7. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $H(d_{ij}) > SG_{\max}$, то $SG_{\max} := H(d_{ij})$; перейти на шаг 14.

Шаг 8. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $V(d_{ij}) < VG_{\min}$, то $VG_{\min} := V(d_{ij})$.

Шаг 9. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $V(d_{ij}) > VG_{\max}$, то $VG_{\max} := V(d_{ij})$; перейти на шаг 14.

Шаг 10. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $H(d_{ij}) < SB_{\min}$, то $SB_{\min} := H(d_{ij})$.

Шаг 11. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $H(d_{ij}) > SB_{\max}$, то $SB_{\max} := H(d_{ij})$; перейти на шаг 14.

Шаг 12. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $V(d_{ij}) < VB_{\min}$, то $VB_{\min} := V(d_{ij})$.

Шаг 13. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $V(d_{ij}) > VB_{\max}$, то $VB_{\max} := V(d_{ij})$; перейти на шаг 14.

Шаг 14. $j := j + 1$.

Шаг 15. Если $j \leqslant w$, то перейти на шаг 2.

Шаг 16. $i := i + 1$; $j := 1$.

Шаг 17. Если $i \leqslant h$, то перейти на шаг 2.

Шаг 18. $TSR := TS(SR)$; $TVR{:=}TV(VR)$; $TSG := TS(SG)$; $TVG{:=}TV(VG)$; $TSB{:=}TS(SB)$; $TVB{:=}TV(VB)$; $i := 1$; $j := 1$.

Шаг 19. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $S(d_{ij}) < TSR$ и $V(d_{ij}) > TVR$, то $p_{ij} := 0$; перейти на шаг 26.

Шаг 20. Если $H(d_{ij}) \in (HR_{\min}, HR_{\max})$ и $S(d_{ij}) > TSR$ и $V(d_{ij}) < TVR$, то $p_{ij} := 1$; перейти на шаг 26.

Шаг 21. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $S(d_{ij}) < TSG$ и $V(d_{ij}) > TVG$, то $p_{ij} := 0$; перейти на шаг 26.

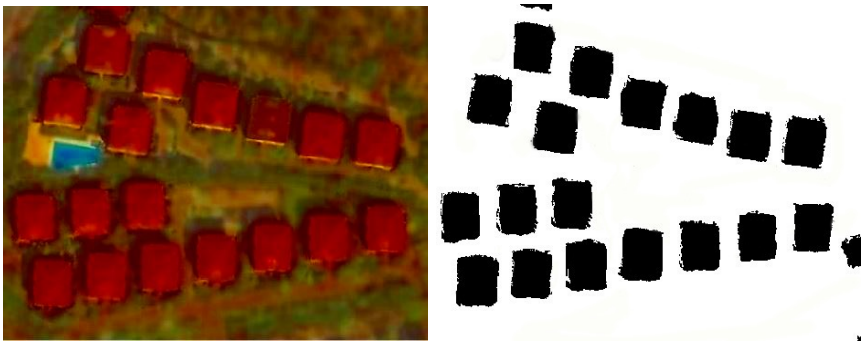
Шаг 22. Если $H(d_{ij}) \in (HG_{\min}, HG_{\max})$ и $S(d_{ij}) > TSG$ и $V(d_{ij}) < TVG$, то $p_{ij} := 1$; перейти на шаг 26.

Шаг 23. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $S(d_{ij}) < TSB$ и $V(d_{ij}) > TVB$, то $p_{ij} := 0$; перейти на шаг 26.

Шаг 24. Если $H(d_{ij}) \in (HB_{\min}, HB_{\max})$ и $S(d_{ij}) > TSB$ и $V(d_{ij}) < TVB$, то $p_{ij} := 1$; перейти на шаг 26.

Шаг 25. $p_{ij} := 1$.
Шаг 26. $j := j + 1$.
Шаг 27. Если $j \leqslant w$, то перейти на шаг 19.
Шаг 28. $i := i + 1$; $j := 1$.
Шаг 29. Если $i \leqslant h$, то перейти на шаг 19.
Шаг 30. Стоп.

На рис. 2 продемонстрирован результат сегментации изображения с помощью модифицированного метода исключения областей. На вход подается изображение в схеме HSV (рис. 2а). В результате работы алгоритма получаем бинарное изображение, представленное на рис. 2б.



а) Изображение в схеме HSV б) Бинаризованное изображение

Рис. 2. Сегментирование изображения методом исключения областей

3.2. Модифицированный метод жука

На втором этапе работы предложенного нами метода распознавания строений выполняется обнаружение объектов и выделение их контуров. Как уже было сказано ранее, для решения данной задачи используется разработанный нами модифицированный алгоритм жука. Метод жука — классический метод контурного анализа, предназначенный для изучения текстурных характеристик и особенностей изображений [13]. Его основная идея заключается в последовательном вычерчивании границы между объектом и фоном в бинаризованном изображении. Области изображения с черным цветом соответствуют объектам поиска, а с белым цветом — фону. Прослеживающая точка в виде «жука» движется по изображению до тех пор, пока не дойдет до темной области (объекта). Тогда «жук» поворачивается налево и движется вперед, пока не достигнет границ объекта, после этого поворачивается направо и повторяет процесс, пока не достигнет окрестности начальной точки. Координаты точек перехода с черного на белое и с белого на черное и описывают границу объекта [14]. Жук может перемещаться по четырем направлениям: влево, вверх, вправо, вниз.

В нашей работе предлагается модифицированная версия алгоритма жука, основанная на «маршруте обхода контура», т.е. списке координат пикселей изображения, посещенных «жуком» при обходе контура. Суть модификации метода состоит в следующем.

1. Для перемещения жука добавлены четыре диагональные направления, т.е. перемещения возможны по восьми направлениям: влево, влево-вверх, вверх, вверх-вправо, вправо, вниз-вправо, вниз, вниз-влево.
2. Добавлен «маршрут обхода контура», представляющий собой список координат точек, посещенных алгоритмом при поиске контура. В процессе обработки изображения вы-

полняется запись координат непроверенного пикселя в «маршрут», чтобы после он стал новой точкой отсчета движения жука. Окрестность нового пикселя верифицируется на наличие пикселей черного цвета — т.е. потенциальных точек для перемещения в них жука. Если пиксель не проходит проверку (т.е. из него никуда нельзя перейти), то он удаляется из «маршрута» жука и проверяется следующая точка (если она есть).

Добавленные диагональные направления позволяют алгоритму лучше справляться с объектами сложной формы, содержащие диагональные линии или изломы (классический алгоритм жука в случае, когда контур объекта содержал последовательность диагональных пикселей, упирался в первую диагональную точку и заканчивал свою работу). Также добавление новых направлений снизило вероятность получения «дыр» (областей белых пикселей) внутри объекта и многократного прохождения по одним и тем же пикселям, что повысило общую эффективность и скорость работы алгоритма. Использование «маршрута обхода контура» позволило алгоритму корректно возвращаться к ранее сохраненным точкам с несколькими вариантами пути и продолжать обход контуров в случае тупиков. Таким образом, алгоритм избегает зацикливания и пропусков участков контура, обеспечивая более точное и полное распознавание объектов.

Введем несколько необходимых определений. Определим множество контуров обнаруженных объектов E следующим образом: $E = \{E_k\}$, где E_k — контур k-го объекта, $E_k = \{(i_t, j_t),\, t = 1, \ldots, m\}$, $i_t \in \{1, \ldots, h\}$, $j_t \in \{1, \ldots, w\}$. Другими словами, E_k — множество точек (пикселей) контура выделенного объекта, где каждая точка задается координатами (i_t, j_t), m — количество точек контура. $SP = \{(i_t, j_t)\}$, $t = 1, \ldots, M$ — «маршрут обхода контура», представляющий собой последовательность координат точек, посещенных алгоритмом при поиске контура. $P = \{p_{ij},\, i = 1, \ldots, h,\, j = 1, \ldots, w\}$ — копия изображения, используемая для работы алгоритма. Здесь h — высота изображения, w — ширина.

В переменной *Dir* хранится значение направления, в котором выполняется перемещение по точкам изображения в конкретный момент работы алгоритма. Приведем перечень допустимых значений, хранимых в данной переменной. Если значение *Dir* равно 1, то выполняется перемещение влево (из пикселя с координатами (i, j) в пиксель с координатами $(i, j-1)$). Если $Dir = 2$, то выполняется перемещение влево-вверх (в пиксель с координатами $(i-1, j-1)$). Значение $Dir = 3$ соответствует перемещению вверх (в $(i-1, j)$). $Dir = 4$ означает перемещение вверх–вправо (в $(i-1, j+1)$), $Dir = 5$ — перемещение вправо (в $(i, j+1)$), $Dir = 6$ — перемещение вниз-вправо, $Dir = 7$ — перемещение вниз, $Dir = 8$ — перемещение вниз-влево. Переменная *Neighbors* предназначена для хранения количества значимых пикселей (черного цвета), находящихся в окрестности текущей точки (с координатами (i, j)). В качестве окрестности точки (i, j) будут рассматриваться точки (t, s), где $t = i-1, \ldots, i+1$, $s = j-1, \ldots, j+1$, $t \neq i$, $s \neq j$.

Опишем этапы работы предлагаемого алгоритма выделения контуров Contours:

Вход алгоритма: D_{binary} — бинарное изображение.

Выход алгоритма: E — множество контуров объектов.

Шаг 1. $E := \varnothing$; $P := D_{\text{binary}}$; $k := 1$; $i := 1$; $j := 1$.

Шаг 2. $E_k := \varnothing$; $SP := \varnothing$; $Dir := 1$.

Шаг 3. Если $(i, j) \notin E$ и $p_{ij} = 1$, то перейти на шаг 9.

Шаг 4. $j := j + 1$.

Шаг 5. Если $j \leqslant w$, то перейти на шаг 3.

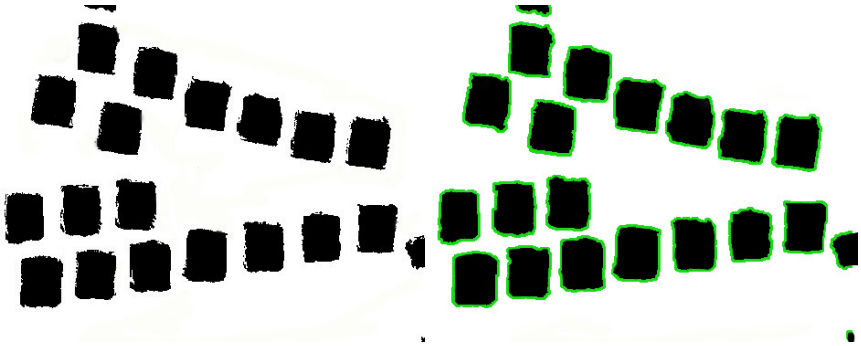
Шаг 6. $i := i + 1$; $j := 1$.

Шаг 7. Если $i \leqslant h$, то перейти на шаг 3.
Шаг 8. Перейти на шаг 34.
Шаг 9. $SP := SP \cup \{(i,j)\}$.
Шаг 10. $Neighbors := \left(\sum\limits_{t=i-1}^{i+1} \sum\limits_{s=j-1}^{j+1} p_{ij} \right) - 1$.
Шаг 11. Если $Neighbors = 0$, то $SP := SP \setminus \{(i,j)\}$; $p_{ij} := 0$; перейти на шаг 4.
Шаг 12. Если $Neighbors < 8$, то $E_k := E_k \cup \{(i,j)\}$.
Шаг 13. Если $Dir = 5$ или $Dir = 6$ или $Dir = 7$ или $Dir = 8$, то перейти на шаг 22.
Шаг 14. Если $j > 1$ и $p_{ij-1} = 1$ и $(i, j-1) \notin SP$, то $j := j - 1$; $Dir := 1$; перейти на шаг 9.
Шаг 15. Если $i > 1$ и $j > 1$ и $p_{i-1j-1} = 1$ и $(i-1, j-1) \notin SP$, то $i := i - 1$; $j := j - 1$; $Dir := 2$; перейти на шаг 9.
Шаг 16. Если $i > 1$ и $p_{i-1j} = 1$ и $(i-1, j) \notin SP$, то $i := i - 1$; $Dir := 3$; перейти на шаг 9.
Шаг 17. Если $i > 1$ и $j < w$ и $p_{i-1j+1} = 1$ и $(i-1, j+1) \notin SP$, то $i := i - 1$; $j := j + 1$; $Dir := 4$; перейти на шаг 9.
Шаг 18. Если $j < w$ и $p_{ij+1} = 1$ и $(i, j+1) \notin SP$, то $j := j + 1$; $Dir := 5$; перейти на шаг 9.
Шаг 19. Если $i < h$ и $j < w$ и $p_{i+1j+1} = 1$ и $(i+1, j+1) \notin SP$, то $i := i+1$; $j := j+1$; $Dir := 6$; перейти на шаг 9.
Шаг 20. Если $i < h$ и $p_{i+1j} = 1$ и $(i+1, j) \notin SP$, то $i := i + 1$; $Dir := 7$; перейти на шаг 9.
Шаг 21. Если $i < h$ и $j > 1$ и $p_{i+1j-1} = 1$ и $(i+1, j-1) \notin SP$, то $i := i+1$; $j := j-1$; $Dir := 8$; перейти на шаг 9.
Шаг 22. Если $j < w$ и $p_{ij+1} = 1$ и $(i, j+1) \notin SP$, то $j := j + 1$; $Dir := 5$; перейти на шаг 9.
Шаг 23. Если $i < h$ и $j < w$ и $p_{i+1j+1} = 1$ и $(i+1, j+1) \notin SP$, то $i := i+1$; $j := j+1$; $Dir := 6$; перейти на шаг 9.
Шаг 24. Если $i < h$ и $p_{i+1j} = 1$ и $(i+1, j) \notin SP$, то $i := i + 1$; $Dir := 7$; перейти на шаг 9.
Шаг 25. Если $i < h$ и $j > 1$ и $p_{i+1j-1} = 1$ и $(i+1, j-1) \notin SP$, то $i := i+1$; $j := j-1$; $Dir := 8$; перейти на шаг 9.
Шаг 26. Если $j > 1$ и $p_{ij-1} = 1$ и $(i, j-1) \notin SP$, то $j := j - 1$; $Dir := 1$; перейти на шаг 9.
Шаг 27. Если $i > 1$ и $j > 1$ и $p_{i-1j-1} = 1$ и $(i-1, j-1) \notin SP$, то $i := i-1$; $j := j-1$; $Dir := 2$; перейти на шаг 9.
Шаг 28. Если $i > 1$ и $p_{i-1j} = 1$ и $(i-1, j) \notin SP$, то $i := i - 1$; $Dir := 3$; перейти на шаг 9.
Шаг 29. Если $i > 1$ и $j < w$ и $p_{i-1j+1} = 1$ и $(i-1, j+1) \notin SP$, то $i := i-1$; $j := j+1$; $Dir := 4$; перейти на шаг 9.
Шаг 30. $t := 1$;
Шаг 31. Если $t > |SP|$, то перейти на шаг 33.
Шаг 32. Для $(i_t, j_t) \in SP$ $p_{i_t j_t} := 0$; $t := t + 1$; перейти на шаг 31.
Шаг 33. $i := 1$; $j := 1$; $k := k + 1$; перейти на шаг 2.
Шаг 34. Стоп.

Результаты работы предлагаемого модифицированного метода жука по нахождению контуров объектов демонстрируются на рис. 3.

3.3. Алгоритм проверки искусственного происхождения объекта

После обнаружения объектов, претендующих на отнесение к зданиям и строениям, необходимо проверить, являются ли они искусственными объектами. Как уже было сказано ранее, искусственно созданные объекты имеют особенности формы: малую постоянную кривизну (объекты круглой формы), либо несколько точек высокой кривизны (объекты



а) Бинаризованное изображение б) Найденные контуры

Рис. 3. Поиск контуров модифицируемым методом жука

прямоугольной формы) или вообще не имеют кривизны. Нами был предложен алгоритм проверки объекта на искусственное происхождение, проверяющий выполнение указанных свойств для каждого объекта, обнаруженного ранее методом жука. Приведем описание данного алгоритма и необходимые определения.

Пусть дано E_k — множество точек (пикселей) контура объекта, где каждая точка задается его координатами (i_t, j_t), а m — количество точек контура. Если поверхность в евклидовом пространстве задана параметрически кусочно C^{-1}-гладкой функцией $r(u, v)$, где параметры u, v изменяются в области E_k на плоскости (u, v), то площадь $Area(E_k)$ находится по следующей формуле: $Area(E_k) = \int\int_{E_k} |r_u \times r_v| du\, dv$, где $\times$ обозначает векторное произведение, а r_u и r_v — частные производные по u и v.

Средняя кривизна области, ограниченной контуром E_k, вычисляется следующим образом: $CU(E_k) = \frac{\oint_{E_k} r ds}{|E_k|}$, где r — радиус локальной кривизны области. Для нахождения радиуса кривизны воспользуемся подходом, используемым в обобщенном преобразовании Хафа, предназначенным для поиска объектов, принадлежащих определенному классу фигур. В данном случае нас будут интересовать фигуры прямоугольной и округлой формы. Главная идея преобразования Хафа для нахождения прямой линии: представление прямой с помощью параметров r и θ, где параметр r — длина радиус–вектора ближайшей к началу координат точки на прямой (нормали к прямой, проведенной из начала координат), а θ — угол между этим вектором и осью абсцисс. Тогда уравнение прямой можно записать $y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{r}{\sin\theta}\right)$ или после преобразования $r = x \cdot \cos\theta + y \cdot \sin\theta$. Пусть прямая задается уравнением $y = kx + b$. Тогда $k = -\frac{\cos\theta}{\sin\theta}$, а $r = \frac{r}{\sin\theta}$. Следовательно, $r = b\sin\theta$, где $\theta = \arctan(-1/k)$. Мы можем найти значения k и b по двум соседним точкам (x_1, y_1), $(x_2, y_2) \in E_k$: $k = (y_2 - y_1)/(x_2 - x_1)$, $b = y_1 - x_1 \cdot ((y_2 - y_1)/(x_2 - x_1))$. Тогда средняя кривизна области, ограниченной контуром E_k, считая ее прямоугольной фигурой (ограниченной прямыми линиями), вычисляется так:

$$CU_{\text{square}}(E_k) = \frac{\sum\limits_{(x_1,y_1)\in E_k} r_{x_1 y_1}}{|E_k|},$$

где $r_{x_1 y_1} = b\sin(\arctan(-1/k))$, где k и b находятся по формулам, указанным выше, а (x_2, y_2) — точка, следующая за (x_1, y_1) во множестве E_k.

Для определения кривизны округлой фигуры воспользуемся идеей, предложенной в преобразовании Хафа для нахождения окружностей. Пространство всех возможных окружностей, проходящих через точку, состоит из трех измерений: координат центра (x, y) и

радиуса возможной окружности R. Уравнение множества всех центров окружностей с радиусом R, проходящих через точку (x_0, y_0): $(x - x_0)^2 + (y - y_0)^2 = R^2$. Если перебрать все точки $(x, y) \in E_k$, для каждой точки найти радиус окружности R, который можно через нее провести, а затем усреднить полученные значения, то будет найдена средняя кривизна области при предположении, что она является округлой фигурой:

$$CU_{\text{circle}}(E_k) = \frac{\sum\limits_{(x,y)\in E_k} R}{|E_k|},$$

где $R = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, а (x_0, y_0) — точка, следующая за (x, y) во множестве E_k.

Нами были вычислены значения функции $CU_{\text{circle}}(E_k)$ для окружности и значения $CU_{\text{square}}(E_k)$ для эталонных фигур прямоугольной формы (прямоугольной, Г-образной, Н-образной и т.д.) Введем переменную α для хранения порогового значения при определении округлости контура и β — пороговое значение при определении прямоугольной формы контура, $\alpha = {}^{19}/_{20}$, $\beta = {}^{9}/_{20}$. Также введем переменную $curv1$ для хранения вычисленного значения средней кривизны области, ограниченной контуром E_k, для прямоугольной формы, переменную $curv2$ — для хранения значения кривизны округлой формы, ϵ — допустимая погрешность при сравнении контуров с эталонами.

Опишем этапы работы предлагаемого алгоритма проверки объекта на искусственное происхождение Artificials:

Вход алгоритма: E — множество контуров объектов.

Выход алгоритма: EA — множество контуров искусственных объектов.

Шаг 1. $k := 1; R := 0; \alpha := {}^{19}/_{20}; \beta := {}^{9}/_{20}$; $EA := E$; $\epsilon := {}^{1}/_{10}$.

Шаг 2. Для $E_k \in E$ $curv1 := CU_{\text{circle}}(E_k)$; $curv2 := CU_{\text{square}}(E_k)$.

Шаг 3. Если $curv1 > \alpha - \epsilon$ и $curv1 < \alpha + \epsilon$, то перейти на шаг 7.

Шаг 4. Если $curv2 > \beta - \epsilon$ или $curv1 < \beta + \epsilon$, то перейти на шаг 7.

Шаг 5. Если $\frac{Area(E \cap E_k)}{Area(E_k)} > {}^{1}/_{20}$, то перейти на шаг 7.

Шаг 6. $EA := EA \setminus \overline{E_k}$.

Шаг 7. $k := k + 1$.

Шаг 8. Если $k \leqslant |E_k|$, то перейти на шаг 2.

Шаг 9. Стоп.

Шаг 5 алгоритма реализует дополнительную проверку для строений, ограниченных прямыми линиями, но не являющихся прямоугольными, чтобы обойти исключение из списка найденных объектов зданий с Г- или Н-образными частями.

3.4. Алгоритм проверки объекта на объемность

Предыдущий алгоритм позволил обнаружить на изображении объекты искусственного происхождения. В рамках нашей задачи требуется находить строения. Очевидно, что существует ряд искусственных плоскостных сооружений, которые при отображении на двумерном рисунке будут похожи на строения (например, парковки автомашин, дороги, линии электропередач и другие). Единственным отличием таких объектов от строений является то, что у них нет объема. Для устранения данной проблемы, нами был разработан алгоритм проверки объекта на то, что у него есть объем. Как уже было сказано выше, признаком того, что объект имеет объем, является наличие у него на спутниковом снимке тени. Помимо указанного способа отличения объемных объектов от плоскостных в алгоритме предполагается использовать функцию выпуклости области C_0, которую можно задать следующим

образом: $C_0(E_k) = \frac{E_k}{Area(E_k)}$, где E_k — множество точек контура объектов, $Area(E_k)$ — площадь фигуры, ограниченной контуром E_k.

Пусть D — исходное изображение, $E_k = \{(i_t, j_t), t = 1, \ldots, m\}$, $i_t \in 1, \ldots, h$, $j_t \in 1, \ldots, w$, RGB — множество оттенков, для которых $R = \{0, \ldots, 55\}$, $G = \{0, \ldots, 55\}$, $B = \{0, \ldots, 55\}$. Для выявления цветовых значений в пикселе воспользуемся функцией $RGB(p_{ij})$, которая возвращает $R(p_{ij})$, $G(p_{ij})$, $B(p_{ij})$. $\overline{D}$ — промежуточное изображение для поиска области тени. Определим функцию $invert(D)$, которая для каждого пикселя изображения D заменяет пиксели в формате RGB на значения $255 - p_{ij}$ в случае, если $p_{ij} \in E_k$. Функция $\max(D_1, D_2)$ вычисляет максимальное значение тона пикселя следующим образом: $\max(RGB(p_{ij}), RGB(\overline{p}_{ij}))$, $p_{ij} \in D$, $\overline{p}_{ij} \in \overline{D}$. Введем переменную *convexity* для хранения вычисленного значения функции $C_0(E_k)$.

Опишем этапы работы предлагаемого алгоритма проверки объекта на объемность VolumeObjects:

Вход алгоритма: EA — множество контуров искусственных объектов.

Выход алгоритма: EV — множество контуров объемных объектов.

Шаг 1. $k := 1$; $EV := \oslash$; $D_{\text{invert}} := invert(D)$; $\overline{D} := \max(D, D_{\text{invert}})$.

Шаг 2. Для $E_k \in EA$ $convexity := C_o(E_k)$.

Шаг 3. Если $convexity \geqslant 0$, то перейти на шаг 8.

Шаг 4. $EV := EV \cup E_k$.

Шаг 5. $k := k + 1$.

Шаг 6. Если $k > |E_k|$, то перейти на шаг 19.

Шаг 7. Перейти на шаг 2.

Шаг 8. $t := 1$.

Шаг 9. Если $t > |E_k|$, то перейти на шаг 5.

Шаг 10. $Neighbors := 0$.

Шаг 11. $i := i_t - 1$; $j := j_t - 1$.

Шаг 12. Для $p_{ij} \in \overline{D}$ $r := R(p_{ij})$, $g := G(p_{ij})$, $b := B(p_{ij})$.

Шаг 13. Если $r \in (0, 55) \cup g \in (0, 55) \cup b \in (0, 55)$, то $Neighbors := Neighbors + 1$.

Шаг 14. $j := j + 1$.

Шаг 15. Если $j \leqslant j_t + 1$, то перейти на шаг 12.

Шаг 16. $i := i_t + 1$; $j := j_t - 1$.

Шаг 17. Если $i \leqslant i_t + 1$, то перейти на шаг 12.

Шаг 18. Если $Neighbors \geqslant \frac{|E_k|}{4}$, то перейти на шаг 4.

Шаг 19. Стоп.

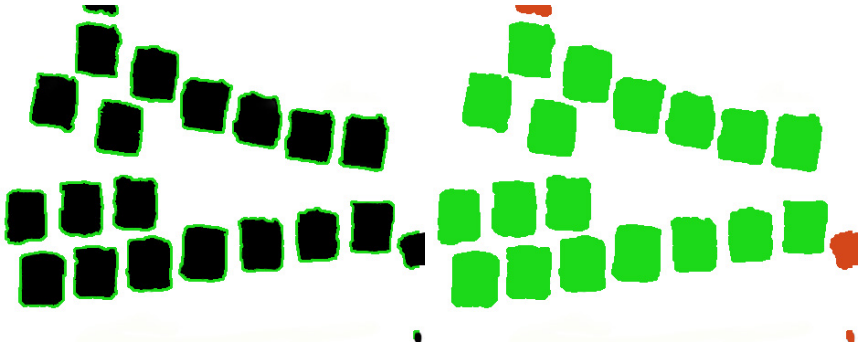
На рис. 4 демонстрируются результаты работы предлагаемого алгоритма проверки объекта на объемность. На рис. 4а приводится изображение с контурами искусственных объектов, подаваемое на вход алгоритму. В результате работы алгоритма получаем изображение с контурами объемных объектов, представленное на рис. 4б.

3.5. Последовательность этапов предложенного гибридного алгоритма распознавания строений

Таким образом, можно записать последовательность этапов работы предложенного гибридного метода распознавания строений следующим образом:

Вход алгоритма: D_{HSV} — цветное изображение в схеме HSV.

Выход алгоритма: EV — множество контуров искусственных объемных объектов.



а) Изображение с контурами искусственных объектов б) Контуры объемных строений

Рис. 4. Выявление искусственных объемных объектов

Шаг 1. $D_{\text{Binary}} = \text{Exclusion}(D_{\text{HSV}})$.
Шаг 2. $E = \text{Contours}(D_{\text{Binary}})$.
Шаг 3. $EA = \text{Artificials}(E)$.
Шаг 4. $EV = \text{VolumeObjects}(EA)$.
Шаг 5. Стоп.

В результате работы алгоритма на изображении будет найдено множество контуров искусственных объемных объектов EV, которое как раз и представляет собой множество контуров строений, а, следовательно, является решением поставленной задачи распознавания строений на спутниковых снимках.

4. Программная реализация

Для решения поставленной в работе задачи распознавания было разработано программное обеспечение «Распознавание строений» на языке Python с использованием библиотек OpenCV, Resterio, Matplotlib, NumPy. Программа состоит из следующих модулей: предобработки изображения, фильтрации, распознавания строений и вывода результата. После загрузки изображения модуль предобработки преобразует изображение к определенному размеру (720×720 пикселей), что позволяет установить на изображении метрическую систему для дальнейшего определения площади объектов. В модуле фильтрации реализованы ряд фильтров и преобразований изображений для улучшения качества изображения, устранения шумов и повышения контрастности: метод сдвига среднего значения, метод скалирования изображения, преобразование изображения в HSV-представление. После этого выполняется модуль распознавания, который реализует предлагаемый в работе гибридный алгоритм распознавания строений. Вид созданного программного обеспечения представлен на рис. 5.

Описанный алгоритм определяет на изображении строения (выделяются на рисунке зеленым цветом). Кроме того, приложение отображает ложно распознанные объекты (искусственные объекты, не прошедшие проверку на объем или слишком малые по площади). Они изображаются на рисунке красным цветом.

5. Результаты проведенных вычислительных экспериментов

В работе были проведены вычислительные эксперименты по оцениванию эффективности предложенного метода, а также его сравнению с тремя известными алгоритмами распознавания. В качестве алгоритмов для сравнения были выбраны классический метод

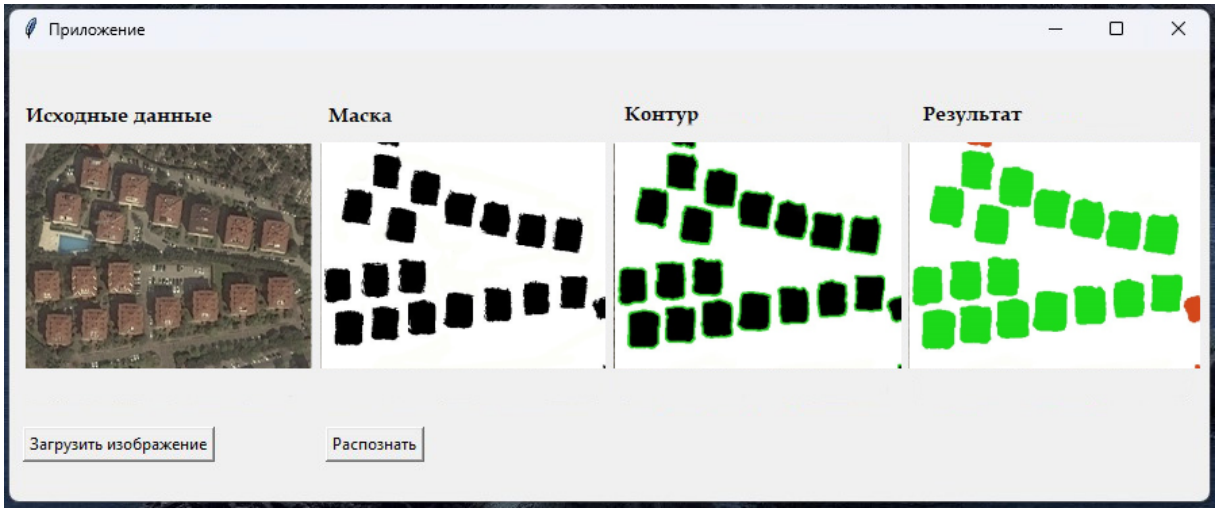


Рис. 5. Вид разработанного приложения

исключения областей [12], метод активных контуров [7] и нейронная сеть с архитектурой UANet + PVT-V2-B2, которая на данный момент является одной из лучших нейросетей обнаружения строений [6]. Сравнение алгоритмов выполнялось на основе известного датасета Inria, представляющего собой набор размеченных спутниковых снимков зданий в формате RGB. Указанный датасет является одним из известных открытых наборов изображений, на которых производят сравнение алгоритмов семантической сегментации строений. Набор данных Inria building содержит 1800 изображений, собранных из пяти городов (Остин, Чикаго, Китсап, Тироль и Ви–Энна), по 360 изображений на каждый город.

Под эффективностью алгоритма будет пониматься качество распознавания и время работы алгоритма. Как уже было сказано в разделе 2.2, в работе в качестве критерия качества распознавания используются четыре метрики: точность, полнота, усредненный индекс Жаккара и мера F1, представляющие собой функции от количества ошибок первого и второго рода [10]. Понятия ошибок первого и второго рода были приведены выше, в разделе 2.2. Под точностью (Precision) понимается доля объектов, действительно принадлежащих данному классу, относительно всех объектов, которые алгоритм отнес к этому классу. Точность вычисляется по формуле $Precision = \frac{TP}{TP+FP}$, где TP — истинно положительные значения, FP — ложно положительные значения, т.е. значения ошибки первого рода. Полнота (Recall) показывает, какую долю объектов класса из всех объектов, действительно относящихся к этому классу, нашел алгоритм: $Recall = \frac{TP}{TP+FN}$, где FN — ложно отрицательные значения (ошибки второго рода). Индекс Жаккара или метрика IoU (Intersection over Union) — метрика степени пересечения между двумя ограничивающими рамками: $IoU = \frac{TP}{TP+FP+FN}$. Мера F1 (F1-Score) — мера, сочетающая точность и полноту: $F1 = \frac{2 \times Precision \times Recall}{Precision+Recall}$.

Для обучения нейросети дополнительно использовалась обучающая выборка из 30000 изображений. Вычислительные эксперименты проводились на компьютере с AMDRyzen 5 2600 Six-CoreProcessor (8 MB Cache, 3.40 ГГц) и ОЗУ объемом 16 Гбайт. В таблице в качестве времени обучения и распознавания приводится среднее время для обработки одного изображения.

Из табл. 1 видно, что нейронная сеть показала наилучшую скорость работы и при этом имеет хорошую точность распознавания. Однако стоит напомнить, что нейронную сеть необходимо предварительно обучить на достаточно объемной выборке. Процесс обучения

Таблица 1. Сравнение работы алгоритмов

Метод	Время, с		Precision	IoU	Recall	F1
	Обуч.	Расп.				
Метод исключения областей [12]	0.1	27	83.35	71.30	82.41	82.87
Метод активных контуров [7]	–	90	**92.10**	**79.93**	**90.85**	**91.47**
UANET [6]	0.12	**5**	86.39	74.40	84.28	85.32
Гибридный алгоритм	–	15	87.28	76.44	85.37	86.31

трудоемок и занимает значительное время. Предложенный нами гибридный алгоритм не требует предварительного обучения и лишь незначительно проигрывает по скорости уже обученной нейросети, что, несомненно, показывает его преимущество. Наилучшую точность распознавания продемонстрировал метод активных контуров, однако он имеет наименьшую скорость работы и чувствительность к качеству изображения. Предложенный же нами метод показал второй результат по точности распознавания при явном выигрыше в скорости работы. Такая же ситуация наблюдается и по остальным метрикам качества распознавания: метод активных контуров занимает первое место, а разработанный нами гибридный метод является вторым.

Таким образом, можно утверждать, что предложенный алгоритм распознавания имеет хорошее быстродействие и высокую эффективность в распознавании зданий на спутниковых снимках. Полученные показатели эффективности работы метода позволяют успешно решать с его помощью задачи мониторинга городской застройки, обнаружения изменений в инфраструктуре, обнаружения незаконно возведенных строений, анализа сельскохозяйственных угодий, исследования лесных ресурсов. Представленные задачи не требуют распознавания в реальном времени и чаще всего решаются в режиме оффлайн (в отличие от ряда задач, имеющих существенные требования к скорости распознавания — например, при распознавании дорожных знаков и препятствий на дороге), поэтому скорость распознавания, продемонстрированная нашим алгоритмом, является достаточной.

Выявленное в результате экспериментов преимущество метода активных контуров в качестве распознавания стало интересным результатом, который требует дальнейшего исследования. К числу известных недостатков данного алгоритма относятся чувствительность к качеству и размеру изображений, поэтому мы планируем провести еще серию экспериментов по сравнению всех методов при первоначальном преобразовании входных изображений больших размеров.

Заключение

В данной статье предложен новый метод распознавания строений на спутниковых снимках, являющийся модифицированным гибридом алгоритма исключения областей и метода жука. Распознавание строений выполняется на цветных спутниковых снимках формата RGB. Перед работой алгоритма распознавания происходит предобработка и фильтрация снимков с помощью метода скалирования изображения, сдвига среднего значения и перевода в цветовую схему HSV. Для решения поставленной задачи предложенный гибридный метод распознавания строений выполняет следующие действия: сегментацию изображения на регионы схожих пикселей; обнаружение объектов и выделение их контуров; выявление среди потенциальных объектов искусственно созданных объектов; обнаружение среди них

объемных объектов. Для сегментации изображения разработан модифицированный метод исключения областей, работающий с цветным изображением в схеме HSV и выполняющий пороговую сегментацию отдельно для каждого цветового тона (красного, зеленого и синего) по яркости и насыщенности. Для обнаружения объектов и выделения контуров в работе предложен модифицированный алгоритм жука, основанный на «маршруте обхода контура». Для перемещения жука добавлены четыре диагональных направления, которые позволяют алгоритму лучше справляться с объектами сложной формы и избегать зацикливания и пропусков участков контура. Использование «маршрута обхода контура» позволило существенно ускорить работу алгоритма. Для выявления искусственно созданных объектов разработан собственный алгоритм проверки контуров объектов на кривизну, а для выявления среди искусственных объектов строений — алгоритм проверки объекта на объемность. Разработано программное обеспечение, реализующее работу предложенного алгоритма распознавания и вышеперечисленных методов предобработки и фильтрации. Проведены вычислительные эксперименты по оцениванию эффективности предложенного метода, а также его сравнению с тремя известными алгоритмами распознавания: классическим методом исключения областей, методом активных контуров и нейронной сетью с архитектурой UANet. Сравнение алгоритмов выполнялось на основе известного датасета Inria. Предложенный гибридный алгоритм показал хорошую скорость и точность распознавания. Одним из самых важных преимуществ разработанного алгоритма является то, что он не требует наличия обучающей выборки и предварительного обучения. Также необходимо отметить автоматическую настройку параметров гибридного алгоритма. В рамках дальнейших исследований мы планируем провести вычислительные эксперименты:

1) по исследованию чувствительности предложенного метода к качеству и разрешению исходных спутниковых изображений;

2) по исследованию чувствительности к качеству и разрешению исходных спутниковых изображений трех перечисленных ранее алгоритмов распознавания;

3) по сравнению алгоритмов на основе других открытых наборов размеченных спутниковых снимков.

Литература

1. Гилин С.В. Задача автоматического распознавания зданий в водоохранных зонах на спутниковых снимках // Информационные технологии и математическое моделирование (ИТММ-2021): Материалы XX Международной конференции имени А.Ф. Терпугова, Томск, 1–5 декабря, 2021. Томск: Национальный исследовательский Томский государственный университет, 2022. С. 6–12.

2. Захаров А.А., Тужилкин А.Ю. Сегментация спутниковых изображений на основе суперпикселей и разрезов на графах // Программные системы и вычислительные методы. 2018. № 1. С. 7–17. DOI: 10.7256/2454-0714.2018.1.25629.

3. Фукунага К. Введение в статистическую теорию распознавания образов. Москва: Наука, 2009. 368 с.

4. Горбачев В.А., Криворотов И.А., Маркелов А.О., Котлярова Е.В. Семантическая сегментация спутниковых снимков аэропортов с помощью сверточных нейронных сетей // Компьютерная оптика. 2020. Т. 44, № 4. С. 636–645. DOI: 10.18287/2412-6179-CO-636.

5. Shahoud A., Shashev D., Shidlovskiy S. Detection of Good Matching Areas Using

Convolutional Neural Networks in Scene Matching-Based Navigation Systems // Ежегодная международная конференция по компьютерной графике и машинному зрению ГрафиКон (GraphiCon-2021): Труды Международной конференции по компьютерной графике и машинному зрению «Графикон», Томск, 27–30 сентября, 2021. Москва: Институт прикладной математики имени М.В. Келдыша РАН, 2021. С. 443–452.

6. He W., Li J., Cao W., Zhang L., Zhang H. Building Extraction from Remote Sensing Images via an Uncertainty-Aware Network. URL: `https://arxiv.org/pdf/2307.12309` (дата обращения: 15.02.2024).
7. Сорокин Д.В., Крылов А.С. Метод активных контуров для сегментации изображений. Москва: ООО «МАКС Пресс», 2022. 116 с.
8. Чернов А.В., Чупшев Н.В. Автоматическое распознавание контуров зданий на картографических изображениях // Компьютерная оптика. 2007. Т. 31, № 4. С. 636–645.
9. Панченко Д.С., Путятин Е.П. Сравнительный анализ методов сегментации изображений // Радиоэлектроника и информатика. 1999. Т. 4. С. 109–114.
10. Воронцов К.В. Математические методы обучения по прецедентам. URL: `http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf` (дата обращения: 10.09.2023).
11. Xu Yu., Wang K., Liu S., Yang S., Yan B. Atmospheric correction of hyperspectral data using MODTRAN model // Remote Sensing of the Environment: 16th National Symposium on Remote Sensing of China. 2008. Vol. 7123. P. 1–7. DOI: 10.1117/12.815552.
12. Голуб Ю.И., Старовойтов В.В., Коноплин Е.Е. Сегментация примерно однородных по яркости областей // Искусственный интеллект. 2008. Т. 3. С. 332–338.
13. Фурман Я.А., Кревецкий А.В., Передреев А.К. Введение в контурный анализ и его приложения к обработке изображений и сигналов. Москва: ФИЗМАТЛИЗ, 2002. 592 с.
14. Сакович И.О., Белов Ю.С. Обзор основных методов контурного анализа для выделения контуров движущихся объектов // Инженерный журнал: наука и инновации. 2014. Т. 36, № 12. С. 1–11.

Баранова Ирина Владимировна, к.ф.-м.н., доцент, базовая кафедра вычислительных и информационных технологий, Институт математики и фундаментальной информатики Сибирского федерального университета (Красноярск, Российская Федерация)

Гилин Степан Валентинович, аспирант, базовая кафедра вычислительных и информационных технологий, Институт математики и фундаментальной информатики Сибирского федерального университета (Красноярск, Российская Федерация)

DOI: 10.14529/cmse240204

BUILDING RECOGNITION HYBRID ALGORITHM FOR SATELLITE IMAGES BASED ON THE BEETLE METHOD AND THE AREA EXCLUSION ALGORITHM

© 2024 I.V. Baranova, S.V. Gilin

Siberian Federal University

(660041 Krasnoyarsk, pr. Svobodny, 79)

E-mail: ibaranova@sfu-kras.ru, gilin.stepan@mail.ru

Received: 10.01.2024

The article proposes a new method for recognizing buildings on satellite images. The proposed method is a hybrid, it is based on the region exclusion algorithm and the beetle method. The region exclusion algorithm is a well-known and effective approach to object detection on the image. Its main idea is to segment an image into regions of similar pixels based on various characteristics: color, texture, brightness, shape, etc. The beetle method is a classic contour analysis method that sequentially draws the boundary between an object and its background. As part of the proposed method, the beetle method first identifies potential areas where buildings may be located. The region exclusion method then eliminates unwanted elements in the image (vegetation, water surfaces and roads) that could be falsely identified as buildings, and accurately determines the location and outline of buildings. The offered algorithm shows good recognition accuracy regardless of image quality and does not require a training sample. The article also describes the software implementation of the proposed method and discusses the results of computational experiments to assess the quality of the method and compare it with three well-known recognition algorithms.

Keywords: building recognition, satellite imagery, hybrid method, beetle method, area exclusion method, textural characteristics, building boundaries, potential areas, classification.

FOR CITATION

Baranova I.V., Gilin S.V. Building Recognition Hybrid Algorithm for Satellite Images Based on the Beetle Method and the Area Exclusion Algorithm. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 56–76. (in Russian) DOI: 10.14529/cmse240204.

References

1. Gilin S.V. The task of automatic recognition of buildings in water protection zones on satellite images. Information Technologies and Mathematical Modeling (ITMM-2021): Proceedings of the XX International Conference named after A.F. Terpugov, Tomsk, Russia, December 1–5, 2021. Tomsk: National Research Tomsk State University, 2022. P. 6–12. (in Russian)
2. Zaharov A.A., Tujilkin A.Y. Segmentation of satellite images based on super pixels and graph sections. Software systems and computational methods. 2018. No. 1. P. 7–17. (in Russian) DOI: 10.7256/2454-0714.2018.1.25629.
3. Fukunaga K. An introduction to the statistical theory of pattern recognition. Moscow: Science, 2009. 368 p. (in Russian)
4. Gorbachev V.A., Krivorotov I.A., Markelov A.O., Kotlyarova E.V. Semantic segmentation of satellite images of airports using convolutional neural networks. Computer optics. 2020. Vol. 44, no. 4. P. 636–645. (in Russian) DOI: 10.18287/2412-6179-CO-636.
5. Shahoud A., Shashev D., Shidlovskiy S. Detection of Good Matching Areas Using Convolutional Neural Networks in Scene Matching-Based Navigation Systems. Annual

International Conference on Computer Graphics and Machine Vision GraphiCon (GraphiCon-2021): Proceedings of the International Conference on Computer Graphics and Machine Vision “Graphicon”, Tomsk, Russia, September 27–30, 2021. Moscow, M.V. Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, 2021. P. 443–452. (in Russian)

6. He W., Li J., Cao W., Zhang L., Zhang H. Building Extraction from Remote Sensing Images via an Uncertainty-Aware Network. URL: `https://arxiv.org/pdf/2307.12309` (accessed: 15.02.2024).
7. Sorokin D.V., Krylov A.S. The active contour method for image segmentation. Moscow: LTD “MAX Press”, 2022. 116 p. (in Russian)
8. Chernov A.V., Chupshev N.V. Automatic detection of building contours in cartographic images. Computer optics. 2007. Vol. 31, no. 4. P. 636–645. (in Russian)
9. Panchenko D.S., Putyanin E.P. Comparative analysis of image segmentation methods. Radio electronics and computer science. 1999. Vol. 4. P. 109–114. (in Russian)
10. Vorontsov K.V. Mathematical methods of teaching by use cases. URL: `http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf` (accessed: 10.09.2023). (in Russian)
11. Xu Yu., Wang K., Liu S., Yang S., Yan B. Atmospheric correction of hyperspectral data using MODTRAN model. Remote Sensing of the Environment: 16th National Symposium on Remote Sensing of China. 2008. Vol. 7123. P. 1–7. DOI: 10.1117/12.815552.
12. Golub Y.I., Starovoytov V.V., Konoplin E.E. Segmentation of areas with approximately uniform brightness. Artificial intelligence. 2008. Vol. 3. P. 332–338. (in Russian)
13. Furman Y.A., Kreveckiy A.V., Peredreev A.K. An introduction to contour analysis and its applications to image and signal processing. Moscow: PHYSICAL ANALYSIS, 2002. 592 p. (in Russian)
14. Sakovich I.O., Belov Y.S. Overview of the main methods of contour analysis for highlighting the contours of moving objects. Engineering Journal: Science and Innovation. 2014. Vol. 36, no. 12. P. 1–11. (in Russian)

УДК 004.4'2, 004.413.5 **DOI: 10.14529/cmse240205**

АНАЛИЗ ИСПОЛНЕНИЯ ФРАГМЕНТИРОВАННЫХ ПРОГРАММ НА ОСНОВЕ ФАКТОРОВ SLOW*

© 2024 С.Е. Киреев[1,2], В.С. Литвинов[2]

[1] *Институт вычислительной математики и математической геофизики СО РАН (630090 Новосибирск, пр. академика Лаврентьева, д. 6),*

[2] *Новосибирский государственный университет (30090 Новосибирск, ул. Пирогова, д. 2)*

E-mail: kireev@ssd.sscc.ru, v.litvinov@g.nsu.ru

Поступила в редакцию: 10.05.2024

При исполнении параллельных программ, основанных на парадигме параллелизма задач, требуется решать ряд проблем, таких как выбор порядка запуска задач с учетом зависимостей между ними, распределение данных и задач по параллельным процессам, балансировка нагрузки на ресурсы. Эти проблемы относятся к области системного параллельного программирования, и их решение, как правило, обеспечивается специальной исполнительной системой. От качества решения этих проблем, а также от структуры и свойств прикладного алгоритма, лежащего в основе параллельной программы, зависит получаемая производительность. Если производительность программы недостаточна, то требуется ее оптимизация, а для этого нужно знать те причины («узкие места»), которые ограничивают ее производительность. Для определения узких мест программы обычно применяется профилирование, т.е. сбор некоторых характеристик исполнения, которые могут указать на источник проблемы. Однако обычные широко используемые средства профилирования параллельных программ не позволяют дать ответ в требуемых понятиях из-за сложности анализа асинхронного исполнения множества задач, а также из-за неспособности выделить в исполняющейся программе прикладную (множество задач) и системную (исполнительная система) компоненты. Поэтому для таких программ требуется разработка новых методов профилирования и анализа. В статье рассматривается проблема получения «понятных» характеристик выполнения параллельных программ на основе параллелизма задач для анализа производительности и оптимизации. Предлагается количественно оценить степень влияния следующих факторов: нехватка работы (Starvation), передача данных (Latency), накладные расходы (Overhead) и конфликт при доступе к общим ресурсам (Waiting for contention resolution). Представлен алгоритм получения соответствующих характеристик для системы фрагментированного программирования LuNA, а также способ их анализа для оптимизации LuNA-программ. Корректность подхода продемонстрирована на ряде синтетических экспериментов. Показано применение подхода к анализу «реальной» программы численного моделирования.

Ключевые слова: анализ производительности, параллельное программирование, фрагментированное программирование, параллелизм задач, система LuNA.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Киреев С.Е., Литвинов В.С. Анализ исполнения фрагментированных программ на основе факторов SLOW // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 77–96. DOI: 10.14529/cmse240205.

Введение

Решение больших численных задач требует использования суперкомпьютеров, а значит, создания эффективных параллельных программ. Существует множество подходов к разработке параллельных программ. Одной из актуальных и часто используемых парадигм, использующихся в параллельном программировании, является параллелизм задач [1]. Эта парадигма предполагает, что параллельная программа описывается как множество задач,

*Статья рекомендована к публикации программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2024».

которые могут выполняться независимо. Кроме того, предполагается, что задачи в рамках одной программы связаны зависимостями по данным и управлению, что ограничивает возможности по их одновременному выполнению. Использование параллелизма задач позволяет достигать высокой эффективности исполнения программ на параллельных вычислительных системах за счет того, что несколько задач могут одновременно выполняться на разных вычислительных устройствах, а обмен данными между вычислительными элементами может происходить на фоне счета. Кроме того, явное выделение независимых частей прикладного алгоритма и явное описание связей между ними позволяет автоматизировать системные функции управления параллельным исполнением задач и реализовать их в виде специальной исполнительной системы. Возможность такой автоматизации особенно важна в случае программирования в модели с распределенной или неоднородной памятью, где к проблеме управления вычислениями добавляется проблема управлением данными. Разделение описания программы на прикладную и системную часть очень удобно, оно позволяет снизить требования к прикладным пользователям суперкомпьютеров и лежит в основе многих средств повышения уровня параллельного программирования.

Ключевыми критериями качества параллельных программ для решения больших численных задач являются производительность, эффективность и масштабируемость. Качественную параллельную программу написать непросто, поэтому существует проблема оптимизации параллельных программ. Для программ, созданных в рамках парадигмы параллелизма задач, проблема оптимизации может быть разделена на две части: оптимизация описания прикладного алгоритма в виде множества взаимосвязанных задач и оптимизация управления задачами в процессе исполнения. Один из способов оптимизации заключается в том, чтобы на основе анализа исполнения параллельной программы выявить ее узкие места и внести в нее соответствующие изменения. Параллельные программы, основанные на параллелизме задач, отличаются большой динамикой исполнения (динамическое создание задач, динамическое отображение задач на вычислительные устройства, взаимодействие задач, наложение по времени и ресурсам исполнения различных частей программы и т.п.). В результате, даже обладая полной информацией об исполнении, трудно выполнить его анализ и выявить причины полученной производительности. Кроме того, многие средства профилирования параллельных программ часто не способны выделить задачи как отдельные единицы исполнения, следовательно, не отражают специфику разделения вычислений на прикладную и системную части. Для понимания причин полученной производительности необходимо получить такие характеристики исполнения, которые давали бы интегральную картину по интересующим аспектам исполнения и учитывали бы специфику используемой парадигмы, в рамках которой создана программа. Как отмечено в работе [2], факторами, ограничивающими масштабируемость параллельных программ, являются:

- Starvation — задержки вследствие недостатка работы, из-за чего не полностью используются вычислительные ресурсы;
- Latency — задержки вследствие доступа к данным, находящимся в другом процессе и требующим передачи через коммуникационную подсистему;
- Overhead — задержки вследствие решения системных задач параллельного программирования, которых не возникает в последовательной программе (управление ресурсами, задачами, данными и т.п.);
- Waiting for contention resolution — задержки вследствие синхронизации доступа к общим ресурсам.

Следуя работе [2], будем далее для обозначения этих факторов использовать аббревиатуру SLOW. Знание того, насколько каждый из этих факторов повлиял на выполнение некоторой параллельной программы, позволит выяснить пути улучшения ее производительности. Для этого необходимо разработать способ количественной оценки влияния этих факторов, а также установить связь каждого из них с конкретными характеристиками параллельной программы.

В данной работе рассматривается подход к анализу исполнения параллельных программ, основанных на парадигме параллелизма задач, на основе количественной оценки факторов SLOW. Подход демонстрируется на примере системы фрагментированного программирования LuNA, реализующей параллелизм задач [3]. Дальнейшее содержимое статьи структурировано следующим образом. Раздел 1 содержит обзор родственных работ. В разделе 2 представлено краткое описание системы LuNA. В разделе 3 рассмотрена связь особенностей LuNA-программ с факторами SLOW. В разделе 4 предложены количественные характеристики SLOW и алгоритм их вычисления в системе LuNA. В разделе 5 представлены эксперименты, подтверждающие работоспособность подхода и демонстрирующие его применение. Заключение содержит резюме результатов, достигнутых в работе, и направления будущих исследований.

1. Обзор родственных работ

В настоящее время существует множество развитых программных средств профилирования и анализа производительности параллельных программ, призванных помочь понять причины полученной производительности и предложить пути оптимизации. Широко известны, например, TAU [4], Score-P [5] и Extrae [6] — средства профилирования и трассировки, Vampir [7], Scalasca [8] и Paraver [9] — средства анализа и отображения результатов профилирования. Также широко используются средства профилирования от Intel, в частности, Intel Trace Analyzer and Collector [10]. Большинство средств поддерживают только наиболее распространенные средства параллельного программирования, такие как MPI и SHMEM для программирования в распределенной памяти, OpenMP и POSIX threads для многопоточного программирования, OpenACC, OpenCL, HIP, CUDA для программирования ускорителей. Соответственно, если их применить для параллельных программ, созданных в другой парадигме, то эти средства не смогут выделить существенные для таких программ характеристики. Поэтому для новых моделей и средств параллельного программирования необходимо разрабатывать специальные подходы к профилированию и анализу производительности.

В области анализа производительности программ на основе параллелизма задач также ведутся активные исследования [11–15]. В работе [14] приводится обзор средств для визуального анализа производительности программ, написанных в парадигме параллелизма задач. Так как в нем делается упор на средства визуализации, большинство рассмотренных в нем работ представляют собой средства графического отображения дерева или графа задач, что, вероятно, слабо применимо для анализа крупномасштабных программ. Некоторый интерес представляет средство TaskInsight [13], которое позволяет собирать различные характеристики исполнения задач и отображать полученную статистику в виде графиков и диаграмм. В частности, авторы используют его для сравнения различных стратегий планирования задач и для изучения влияния на производительность подсистемы памяти. В работе [12] используется подход, близкий к представленному в настоящей статье. Разра-

батываемое авторами средство Delay Spotter позволяет проанализировать исполнение многопоточной программы, выполняющей множество задач, разделяя время рабочих потоков на четыре компонента: 1) поток выполняет задачу, 2) поток не выполняет задачу, но готовые к выполнению задачи есть, 3) готовых для выполнения задач нет из-за планировщика, 4) готовых для выполнения задач нет из-за логики прикладной программы. Имеется возможность получить данные характеристики интегрально по всей программе, в привязке к дереву задач и в развертке по времени, что дает детальное объяснение полученной производительности. По сравнению с работой [12] подход, предлагаемый в настоящей работе, ориентирован на использование в системах с распределенной памятью, хотя лежащая в основе идея аналогична.

2. Система фрагментированного программирования LuNA

Система фрагментированного программирования LuNA [3] разрабатывается в ИВМиМГ СО РАН в рамках развития технологии фрагментированного программирования (ТФП) [16]. ТФП нацелена на автоматизацию создания эффективных параллельных реализаций численных алгоритмов для суперкомпьютеров. Ключевыми аспектами ТФП являются: явное разделение в программе описания прикладного алгоритма и системной части, отвечающей за исполнение алгоритма; представление прикладного алгоритма в явнопараллельной форме, ориентированной на автоматизацию обеспечения нефункциональных свойств. В соответствии с ТФП предполагается, что прикладной алгоритм должен быть фрагментирован, т.е. представлен в виде множества фрагментов данных и фрагментов вычислений, связанных информационными зависимостями. Таким образом, ТФП не только поддерживает парадигму параллелизма задач (фрагментацию вычислений), но и предусматривает явную фрагментацию данных. Помимо системы LuNA, похожую модель представления алгоритма для распределенных вычислений используют системы PARSEC [17], OCR [18] и Legion [19].

Система LuNA включает язык LuNA и средства исполнения программ на этом языке (LuNA-программ). LuNA-программа представляет собой компактную запись потенциально бесконечного ориентированного графа потока данных с вершинами двух видов — переменные единственного присваивания и операции однократного срабатывания. Дуги графа задают информационные зависимости между переменными и операциями (отношения входа и выхода). Этот граф будем называть графом программы. В терминологии ТФП переменные называются фрагментами данных, а операции — фрагментами вычислений. Каждая операция реализуется некоторым фрагментом кода. Различаются атомарные фрагменты кода (подпрограммы на языке C++) и структурированные фрагменты кода (подпрограммы на языке LuNA). Таким образом, фрагмент вычислений — это фрагмент кода, примененный к конкретным входным и выходным фрагментам данных. Язык LuNA поддерживает условные фрагменты вычислений для описания ветвящихся алгоритмов, индексированные имена фрагментов данных и вычислений, чтобы оперировать массивами фрагментов, операторы for и while для компактной записи множеств фрагментов. Далее будем считать, что граф программы содержит только атомарные фрагменты вычислений, рассматривая структурированные фрагменты кода как компактную запись некоторого подграфа программы. Потенциальная бесконечность графа программы является следствием заранее неизвестного (вычисляемого) количества фрагментов, описываемых операторами for и while, а также

заранее неизвестной (вычисляемой) глубины рекурсии структурированных фрагментов вычислений.

Процесс выполнения LuNA-программы предполагает запуск конкретных фрагментов вычислений в порядке, соответствующем информационным зависимостям, при этом конкретные фрагменты данных получают свои значения, и выбираются конкретные ветви исполнения. Таким образом, реализуется некоторый конечный подграф исходного потенциально бесконечного графа программы. Этот конечный подграф будем называть графом исполнения.

В ходе развития проекта было разработано несколько реализаций системы LuNA. Описываемая в данной работе реализация упоминается в работах [20–22] и основана на исполнительной системе, построенной на основе MPI и C++ threads. Данная исполнительная система обеспечивает динамическое отображение фрагментов данных и вычислений на процессы в соответствии с заданным способом распределения. При этом каждому фрагменту данных назначается номер процесса, где он должен храниться, а фрагменту вычислений назначается номер процесса, где он должен выполниться. Для исполнения фрагментов вычислений в каждом процессе используется пул рабочих потоков. Исполнительная система поддерживает динамическую балансировку нагрузки путем динамического изменения отображения фрагментов на процессы. В работах [23, 24] представлено сравнение рассматриваемой здесь исполнительной системы со следующей версией системы LuNA.

3. Факторы SLOW в системе LuNA

Во введении были рассмотрены факторы SLOW, ограничивающие масштабируемость, а значит, производительность и эффективность параллельных программ. В данном разделе мы установим связь этих факторов с процессом исполнения LuNA-программ, рассмотрим, какие особенности LuNA-программ и исполнительной системы приводят к преобладанию того или иного фактора, и какими способами влияние этих факторов может быть уменьшено.

3.1. Факторы SLOW в процессе исполнения LuNA-программ

Рассмотрим особенности исполнения фрагментов вычислений в распределенной исполнительной системе LuNA. Фрагмент вычислений готов к запуску только тогда, когда все его входные фрагменты данных вычислены и находятся в том же процессе, что и сам фрагмент вычислений. В таком случае этот фрагмент вычислений формирует задачу, которая помещается в очередь задач, потребляемых пулом рабочих потоков данного процесса. Если входные фрагменты данных не вычислены, то рассматриваемый фрагмент вычислений не готов к запуску, т.е. он не формирует задачу. В принципе, он может никогда не стать готовым, если это предусматривается логикой программы. Когда фрагменты вычислений не становятся готовыми по причине отсутствия вычисленных входных фрагментов данных, и при этом происходит простой рабочих потоков, тогда такая ситуация называется «голодание» (фактор Starvation).

Если в некоторый момент времени входные фрагменты данных вычислены, но находятся в другом процессе, то известно, что рассматриваемый фрагмент вычислений в будущем гарантированно сформирует задачу, которая будет запущена, но пока он вынужден ждать получения данных. Это вынужденное ожидание, связанное с передачей фрагментов данных к месту назначения и приводящее к простою рабочих потоков, является фактором Latency.

Кроме исполнения фрагментов вычислений и ожидания по различным причинам, вычислительные устройства (ядра) исполняют код исполнительной системы. Эта работа необходима, но она также является сдерживающим фактором (фактором Overhead), поскольку занимает вычислительные ресурсы.

Что касается фактора Waiting for contention resolution, то, вообще говоря, модель исполнения LuNA-программы не предполагает наличия общих ресурсов благодаря тому, что фрагменты данных записываются единожды. Но некоторая конкретная реализация исполнительной системы может использовать такие общие ресурсы, например, переиспользуемые участки памяти для хранения фрагментов данных. Рассматриваемая в данной работе реализация исполнительной системы LuNA общих ресурсов не использует, поэтому далее будем считать, что на LuNA-программу данный фактор не действует, и рассматривать его мы не будем.

3.2. Связь особенностей LuNA-программ и исполнительной системы с факторами SLOW

Рассмотрим для каждого фактора, какие особенности LuNA-программ и работы исполнительной системы приводят к его чрезмерному влиянию, а также какими способами это влияние может быть уменьшено.

- Starvation — недостаток готовых фрагментов вычислений для рабочих потоков.
  - Информационные зависимости между фрагментами вычислений могут приводить к тому, что степень параллелизма в программе оказывается недостаточно высокой, чтобы загрузить все вычислительные устройства (программа «слишком последовательная»). Причиной может быть как низкая степень фрагментации алгоритма, т.е. данные и вычисления декомпозированы на недостаточное количество частей, так и свойства исходного прикладного алгоритма, например, если прикладной алгоритм принципиально последовательный.
  - Плохое распределение фрагментов вычислений по процессам может приводить к тому, что готовые фрагменты вычислений скапливаются на одних процессах, в то время как рабочие потоки других процессов простаивают. Следует изменить распределение фрагментов вычислений по процессам или применить динамическую балансировку нагрузки.
  - Неверный порядок выбора готовых фрагментов вычислений для исполнения может приводить к тому, что фрагменты вычислений на критическом пути алгоритма будут выбираться в последнюю очередь — только тогда, когда для других рабочих потоков работы уже не осталось. Фрагменты вычислений на критическом пути следует запускать на исполнение в приоритетном порядке.
- Latency — ожидание передачи фрагментов данных.
  - Причиной сильного влияния этого фактора может быть плохое взаимное распределение фрагментов данных и фрагментов вычислений по процессам, т.е. либо фрагменты данных вырабатываются далеко от тех процессов, в которых они должны храниться, либо они хранятся далеко от тех фрагментов вычислений, которые их потребляют. Общее правило состоит в том, что фрагменты, непосредственно связанные дугами в графе алгоритма, следует размещать в одном и том же или близко расположенных процессах.

  - Затраты на ожидание данных иногда можно уменьшить путем разбиения фрагментов на более мелкие (увеличения степени фрагментации), что позволяет передачу мелких фрагментов данных (частей изначально большого фрагмента данных) совместить с их обработкой.
  - Уменьшить количество передач данных между процессами также можно с помощью алгоритмов дублирования и кэширование фрагментов данных, которые могут быть реализованы в исполнительной системе.
- Overhead — работа исполнительной системы.
  - Слишком мелкая фрагментация алгоритма порождает большое количество фрагментов, и исполнительной системе приходится тратить время на управление ими. Уменьшение степени фрагментации может уменьшить накладные расходы.
  - На величину накладных расходов влияет способ реализации исполнительной системы. Например, часть решений по организации исполнения может быть перенесена на этап компиляции программы, как это сделано в последующей версии системы LuNA [23, 24]. Для некоторых классов алгоритмов возможно использование специализированных исполнительных систем, в которых накладные расходы сведены к минимуму [25, 26].
  - В тех случаях, когда одну и ту же LuNA-программу требуется исполнять многократно с разными входными данными, но при условии неизменности графа исполнения, возможно применение техники «проигрывания трассы» для существенного уменьшения накладных расходов [27]. Данная техника предполагает, что при первом запуске LuNA-программы сохраняется трасса исполнения, содержащая информацию о том, в каких процессах и в каком порядке исполнялись фрагменты вычислений. При последующих запусках этой же LuNA-программы сложная исполнительная система заменяется простой, которая только запускает фрагменты вычислений в соответствии с имеющейся трассой. Более того, так как трасса содержит информацию о длительности исполнения фрагментов вычислений, появляется возможность спланировать более эффективное распределение фрагментов по процессам [28].

4. Количественная оценка факторов SLOW в системе LuNA

4.1. Характеристики SLOW

Для количественной оценки факторов SLOW введем характеристики SLOW: P_S, P_L, P_O, P_W. Это интегральные характеристики исполнения программы, представляющие собой долю процессорного времени (в процентах от общего), затраченного в результате действия каждого из этих факторов. Общее затраченное процессорное время (T_{total}) определим как время работы программы по таймеру системного времени (T_{wall}), умноженное на количество задействованных вычислительных устройств (ядер). Например, если программа работала две секунды и занимала четыре ядра, то T_{total} равно восемь секунд. Дополнительно имеет смысл рассмотреть характеристику P_U — долю процессорного времени, затраченного на полезные вычисления (Useful work). Для введенных характеристик должно выполняться соотношение: $P_S + P_L + P_O + P_W + P_U = 100\%$. Значения характеристик SLOW показывают степень влияния того или иного фактора на полученную производительность рассматриваемой параллельной программы на данной вычислительной системе. Если какие-то факторы

имеют особенно большое влияние, то именно на них должны быть направлены усилия по оптимизации.

Идея способа измерения характеристик SLOW состоит в том, чтобы в общем времени исполнения параллельной программы выделить периоды полезной работы, а остальное время разделить на части в соответствии с факторами SLOW. Введем соответствующий набор временны́х характеристик и покажем, чему они соответствуют в применении к исполнению LuNA-программ:

- T_S — время простоя рабочих потоков, связанное с отсутствием готовых к исполнению фрагментов вычислений;
- T_L — время простоя рабочих потоков, связанное с ожиданием получения готовых фрагментов данных, являющихся входными для фрагментов вычислений;
- T_O — время, затраченное на работу исполнительной системы;
- T_W — время простоя рабочих потоков, связанное с тем, что некоторый фрагмент вычислений ожидает освобождения некоторого общего ресурса (в рамках его процесса), занятого другим фрагментом вычислений или исполнительной системой;
- T_U — время работы рабочих потоков, затраченное на исполнение фрагментов вычислений.

Для временных характеристик выполняется соотношение: $T_S + T_L + T_O + T_W + T_U = T_{total}$. Характеристики SLOW тогда будут определяться так: $P_X = T_X / T_{total} \times 100\%$, где X — одно из S, L, O, W, U. Как было отмечено выше, у фрагментов вычислений LuNA-программы нет общих ресурсов, которые могут вызвать ожидание, поэтому $T_W = 0$ и $P_W = 0$. Новый набор факторов и характеристик (с добавлением U и без W) будем обозначать аббревиатурой SLOU.

4.2. Вычисление характеристик SLOU в процессе исполнения LuNA-программы

Для вычисления временных характеристик необходимо выполнить профилирование LuNA-программы, собирая в процессе исполнения информацию о следующих событиях: запуск фрагмента вычислений, завершение фрагмента вычислений, отправка фрагмента данных от одного процесса другому, получение процессом фрагмента данных. При этом требуется также отмечать сопутствующую информацию о событиях: время по системному таймеру, время по таймеру потока, номер процесса, номер рабочего потока, идентификаторы фрагментов данных и фрагментов вычислений.

Как известно, исполнение потоков может прерываться другими процессами и потоками этой же параллельной программы, других программ и операционной системы. Кроме того, операционная система может переносить поток с одного ядра на другое. В таких условиях становится сложно получить адекватные временные оценки исполнения интересующих нас частей программы. Поэтому ограничим условия запуска: будем считать, что рабочие потоки привязаны к ядрам, а потоки исполнительной системы — не обязательно привязаны и могут прерывать рабочие потоки. Кроме того, пусть разные процессы параллельной программы занимают различные непересекающиеся подмножества ядер. Данные условия могут быть удовлетворены при соответствующей настройке исполнительной системы и параметров запуска параллельной программы. Влияние других программ и операционной системы на работу параллельной программы будем считать пренебрежимо малым, хотя в реальности этого и не всегда удается достичь.

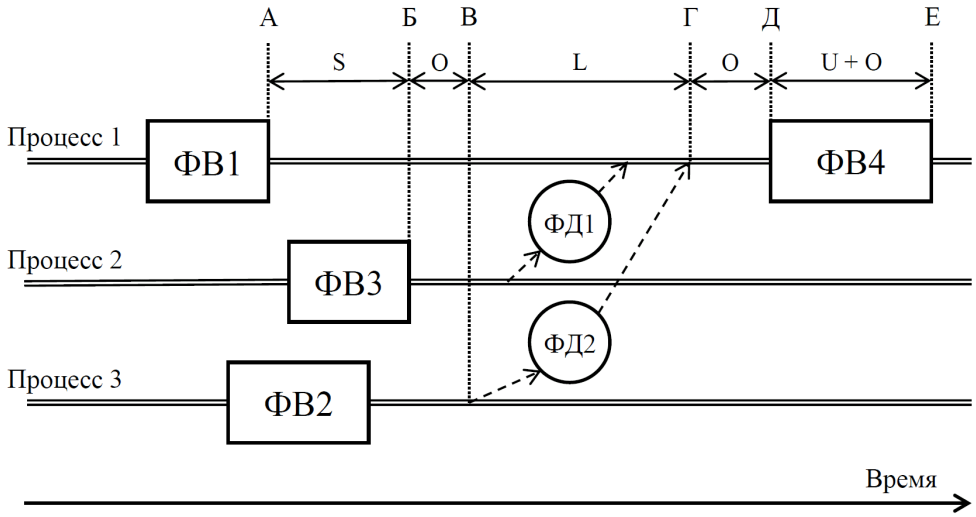


Рис. 1. Временная диаграмма исполнения LuNA-программы. Буквами S, L, O, U обозначены факторы, влияющие на продолжительность данного периода времени: Starvation, Latency, Overhead, Useful work

Опишем алгоритм вычисления временных характеристик, используя конкретный пример. На рис. 1 приведена часть временной диаграммы исполнения LuNA-программы, состоящей из трех процессов, имеющих по одному рабочему потоку. На этом временном отрезке процесс 1 выполнил фрагменты вычислений ФВ1 и ФВ4, процесс 2 — ФВ3, а процесс 3 — ФВ2. Кроме того, ФВ4 принимает на вход фрагменты данных ФД1 и ФД2, выработанные соответственно фрагментами вычислений ФВ3 и ФВ2 (другие информационные зависимости не показаны). Буквами А-Е обозначены интересующие нас моменты времени.

Чтобы получить время полезной работы T_U, нужно измерить и просуммировать времена исполнения всех фрагментов вычислений. Например, на рис. 1 время исполнения ФВ4 обозначено отрезком ДЕ. Однако исполнение фрагмента вычислений может прерываться потоками исполнительной системы. Поэтому для определения реально затраченного полезного времени (вклад в характеристику T_U) необходимо замерить продолжительность этого интервала по таймеру времени потока, а вычтя это время из продолжительности интервала ДЕ по таймеру системного времени, мы получим вклад в характеристику T_O.

Далее необходимо проанализировать все временные интервалы, когда рабочие потоки не были заняты полезной работой. Для примера рассмотрим анализ интервала АД для единственного потока процесса 1 на рис. 1. Часть этого времени данный поток выполнял код исполнительной подсистемы, а часть — находился в ожидании из-за факторов Starvation и Latency, — это те причины, по которым ФВ4 не запустился раньше. Можно было бы сказать, что во время интервала АД рабочий поток простаивал не только из-за неготовности ФВ4, но и из-за неготовности каких-то других фрагментов вычислений, которые могли бы его «обогнать» и запуститься раньше. Почему же мы рассматриваем причины, сдерживавшие запуск именно ФВ4? Потому что именно он из всех кандидатов оказался «наиболее близок» к готовности и запустился в данное время в данном потоке. При другом исполнении этой же LuNA-программы другой фрагмент вычислений мог бы «обогнать» ФВ4 и запуститься в этом потоке, и тогда бы мы анализировали причины задержки другого фрагмента вычислений.

В соответствии с логикой, изложенной в предыдущем разделе, разделим интервал АД следующим образом. Рассмотрим все фрагменты вычислений, которые вырабатывают фрагменты данных, входные для ФВ4. Это фрагменты вычислений ФВ2 и ФВ3. Определим момент времени, когда завершился последний из них — это момент Б. С этого момента все входные фрагменты данных для ФВ4 готовы, и он потенциально уже может запуститься. Значит, в интервале АБ ФВ4 не был готов из-за факторов Starvation и Overhead, а в интервале БД — из-за факторов Latency и Overhead. Попробуем теперь выделить из этих интервалов фактор Overhead.

Если нам известны моменты отправки и приема ФД1 и ФД2, то мы можем определить, когда процессом 1 получен последний из них. Это момент Г, когда был доставлен ФД2. С этого момента запуск ФВ4 откладывался только из-за накладных расходов на работу системных алгоритмов, значит, период ГД дает вклад в характеристику T_O. Кроме того, ФД2 был вычислен процессом сразу после завершения ФВ2, но был отправлен только в момент В. Причиной этой задержки также будем считать работу системных алгоритмов. Пересечение этой задержки с интервалом БД будет интервал БВ, который тоже дает вклад в характеристику T_O. Оставшийся интервал ВГ будем относить к фактору Latency, т.е. вклад в характеристику T_L. Даже если в этом интервале исполнение рабочего потока прерывалось потоками исполнительной системы, то все равно не имеет смысла учитывать эти события, т.к. их возможное отсутствие не приблизило бы момент получения ФД2. Также и интервал АБ мы полностью отнесем к фактору Starvation (вклад в характеристику T_S) по той причине, что исключение возможных прерываний рабочего потока процесса 1 не приблизит момент завершения ФВ3.

Итак, анализируя рассмотренным выше способом все время функционирования всех рабочих потоков и суммируя все вклады, мы получаем характеристики T_S, T_L, T_O и T_U. При использовании нескольких рабочих потоков в рамках одного процесса принцип подсчета характеристик остается тем же. При этом считается, что время передачи фрагмента данных от одного потока к другому в рамках одного процесса (вклад в характеристику T_L) равно нулю. Если исполнительная система дополнительно использовала другие ядра вычислительной системы кроме тех, к которым привязаны рабочие потоки, то все время использования этих ядер следует прибавить к характеристике T_O.

5. Вычислительные эксперименты

Представленный выше алгоритм вычисления временных характеристик T_S, T_L, T_O и T_U был реализован в исполнительной системе LuNA. В данной реализации информация о нужных событиях в ходе исполнения LuNA-программы сохраняется в файл, а после завершения LuNA-программы специальная утилита анализирует собранную информацию и вычисляет временные характеристики SLOU. Ниже представлены эксперименты двух видов. Первые три подраздела содержат синтетические эксперименты, нацеленные на проверку работоспособности предложенного подхода. Каждый из этих экспериментов основан на специальным образом сформированной LuNA-программе с некоторым параметром, от значения которого зависит степень влияния заданного фактора. Эксперименты показывают, что получаемые значения временных характеристик действительно меняются так, как предполагалось. Это подтверждает, что характеристики LuNA-программы действительно связаны с оцениваемыми временными характеристиками исполнения ожидаемым образом. В последнем подразделе представлен анализ исполнения LuNA-программы, решающей «реальную» за-

Рис. 2. Зависимость временных характеристик от степени параллелизма

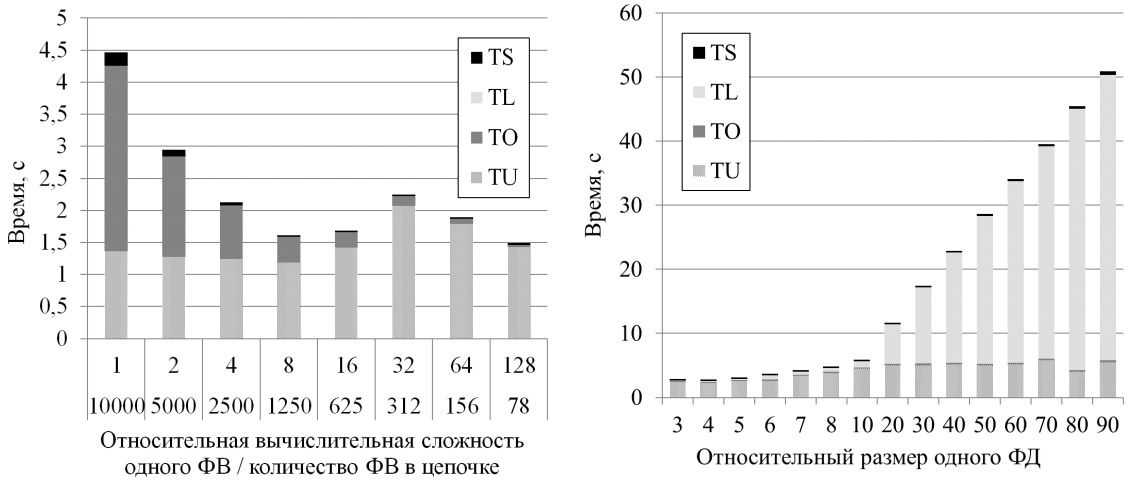
дачу численного моделирования. Все эксперименты проводились на кластере МСЦ РАН [29] МВС-10П Торнадо. Узлы кластера содержат два 8-ядерных процессора Intel Xeon E5-2690 2.9 ГГц (16 ядер на узел) и два ускорителя Intel Xeon Phi 7110X (не использовались в экспериментах).

5.1. Влияние фактора Starvation

Цель эксперимента — продемонстрировать влияние степени параллелизма фрагментированного алгоритма на характеристику T_S. LuNA-программа представляет собой N независимых цепочек ФВ-ФД-ФВ-ФД-...-ФВ постоянной длины (100 фрагментов вычислений в одной цепочке) с одинаковой вычислительной нагрузкой для всех фрагментов вычислений. Фрагменты вычислений внутри каждой цепочки вынуждены исполняться последовательно из-за информационных зависимостей. Вычислительная сложность всех фрагментов вычислений бралась одинаковой, обратно пропорциональной N, чтобы суммарная вычислительная сложность программы сохранялась постоянной. Выполнялась серия запусков программы с количеством цепочек N, изменяющимся от 1 до 16. Запуски производились на одном 16-ядерном узле с одним процессом и 8-ю рабочими потоками. Ожидаемый результат эксперимента — постепенное уменьшение характеристики T_S почти до нуля с ростом N от 1 до 8 в связи с увеличением степени параллелизма LuNA-программы и занятием простаивающих рабочих потоков. При значениях N от 8 до 16 ожидалось пренебрежимо малое значение характеристики T_S (накапливаемое в момент завершения работы программы). Результаты эксперимента, представленные на рис. 2, демонстрируют ожидаемое поведение характеристики T_S. Поведение характеристики T_O также понятно — его значение немного увеличивается с ростом числа фрагментов. Значение характеристики T_L равно нулю, поэтому на рис. 2 ее не видно. Вопросы вызывает только изменение характеристики T_U, особенно скачок при $N = 6$.

5.2. Влияние фактора Overhead

Цель данного эксперимента — продемонстрировать изменение доли времени, которое тратится на работу исполнительной системы при изменении степени фрагментации задачи. LuNA-программа представляет собой одну цепочку ФВ-ФД-ФВ-ФД-...-ФВ с одинаковой вычислительной нагрузкой для всех фрагментов вычислений — по сути, последователь-



(а) (б)

Рис. 3. Зависимость временных характеристик от параметров фрагментированной программы: а) степени фрагментации, б) объема передаваемых данных

ная программа. Запуски производились на одном процессе с одним рабочим потоком. Была выполнена серия запусков с постепенно уменьшающейся степенью фрагментации. При каждом последующем запуске вычислительная нагрузка всех фрагментов вычислений удваивалась, а длина цепочки фрагментов сокращалась в два раза, чтобы общая вычислительная сложность LuNA-программы оставалась неизменной. От данного эксперимента ожидалось уменьшение характеристики T_O с уменьшением степени фрагментации. Результаты, представленные на рис. 3а, в целом соответствуют ожиданиям. Как и в прошлом эксперименте, здесь остается непонятным изменение характеристики T_U при том, что вычислительная сложность программы должна была оставаться постоянной. Значение характеристики T_L здесь также равно нулю, и на рис. 3а ее не видно.

5.3. Влияние фактора Latency

Цель данного эксперимента — исследовать изменение характеристики T_L при изменении объема передаваемых данных между процессами. Для этого LuNA-программа запускалась на двух процессах на разных узлах кластера, в каждом из которых работало по одному рабочему потоку. Сама LuNA-программа состоит из двух цепочек А и Б вида ФВ-ФД-ФВ-ФД-...-ФВ равной длины. В цепочке А каждый четный фрагмент вычислений назначается на первый процесс, а каждый нечетный — на второй. В цепочке Б, наоборот, каждый четный фрагмент вычислений назначается на второй процесс, а каждый нечетный — на первый. Таким образом, каждый вычисленный в некотором процессе фрагмент данных должен быть передан в другой процесс. Вычислительная нагрузка для каждого фрагмента вычислений была выбрана одинаковой для того, чтобы уменьшить значение характеристики T_S в предположении, что интервалы времени, в течение которых работают i-ые фрагменты вычислений из разных цепочек, примерно совпадают. Эксперимент состоит из серии запусков. В каждом запуске размер всех фрагментов данных совпадает, но растет от запуска к запуску. Предполагается, что T_L будет увеличиваться при увеличении размера фрагментов данных. Результаты эксперимента, представленные на рис. 3б, соответствуют ожиданиям.

5.4. Анализ LuNA-реализации PIC-метода

В данном подразделе представлен анализ «реальной» задачи численного моделирования на основе факторов SLOU. В качестве «реальной» задачи использовалась реализованная в системе LuNA задача трехмерного моделирования самогравитирующего пылевого облака методом частиц-в-ячейках (PIC-методом) [30]. Реализация этой задачи в системе LuNA упоминалась в [25]. Для эксперимента были взяты следующие параметры:

- размеры пространственной сетки: $100 \times 100 \times 100$,
- количество модельных частиц: 1 000 000,
- число шагов по времени: 10,
- фрагментация области моделирования — количество подобластей, на которые она разбивается по каждому направлению: $4 \times 4 \times 4$.

В табл. 1 представлены полученные результаты для двух запусков задачи на разных ресурсах: одном и четырех узлах кластера, по одному процессу (16 рабочих потоков) на узел.

Таблица 1. Характеристики SLOU двух вариантов запуска задачи на разных ресурсах

	T_{wall}	T_{SLOU} (P_{SLOU})	T_S (P_S)	T_L (P_L)	T_O (P_O)	T_U (P_U)
1 узел × 16 потоков	9.99 с	153.8 с (96.25%)	30.1 с (18.84%)	0 с (0%)	110.6 с (69.21%)	13.1 с (8.20%)
4 узла × 16 потоков	164.05 с	10471 с (99.73%)	7669 с (73.05%)	1556 с (14.82%)	1234 с (11.75%)	12 с (0.11%)

Из полученных результатов можно сделать следующие выводы.

- При сравнении значений T_{wall} для двух разных запусков задачи видно, что, несмотря на увеличение общего количества доступных ядер в четыре раза, время счета задачи увеличивается почти в 16 раз. Подобное замедление на данной версии исполнительной системы LuNA наблюдалось и на других задачах [20].
- Сумма полученных значений временных характеристик (T_{SLOU}) составляет более 96% от общего затраченного процессорного времени T_{total}, что вполне достаточно для определения основных проблем в реализации. С другой стороны, незначительная величина процессорного времени осталась неучтенной, что говорит о том, что реализацию алгоритма определения характеристик еще можно улучшать.
- Характеристика P_O даже при запуске на одном узле составляет около 69%, что свидетельствует о высоких накладных расходах на работу исполнительной системы. Кроме того, при использовании нескольких узлов характеристика T_O вырастает более чем в 10 раз. Для ее снижения, вероятно, имеет смысл оптимизировать исполнительную систему или переписать ее на других принципах (что и было сделано, см. [23, 24]). Используя же текущую версию, можно попробовать уменьшить степень фрагментации задачи.
- При запуске задачи на нескольких узлах кластера значение характеристики P_S составило около 73%, т.е. фактор Starvation в данном случае является главной причиной низкой производительности. Это свидетельствует о том, что либо для решения данной задачи не требуется использовать так много вычислительных ресурсов, либо наблюдается дисбаланс нагрузки. Эффективность использования ресурсов, вероятно, можно

увеличить, увеличив степень фрагментации задачи или включив динамическую балансировку нагрузки.

- Время полезной работы T_U для двух запусков немного отличается, хотя задача решалась одна и та же. Подобная разница в значениях T_U наблюдалась и в синтетических экспериментах. Причины этого еще предстоит выяснить.

Заключение

В работе предложен способ анализа исполнения фрагментированных программ на основе сравнительной оценки влияния факторов SLOW. Показано, как данные факторы связаны с различными характеристиками фрагментированных программ и исполнительной системы. Предложены количественные характеристики SLOW, отражающие степень влияния факторов SLOW на производительность. Представлен алгоритм вычисления количественных характеристик для одной реализации системы LuNA. Применимость подхода для анализа производительности LuNA-программ подтверждается экспериментами.

Предложенный подход может дополнить существующие способы анализа производительности параллельных программ, основанных на параллелизме задач. В частности, данный подход может применяться и к другим системам параллельного программирования, основанным на данной парадигме [17–19], и быть интегрирован в специализированные средства анализа производительности для этих систем.

Продолжением данной работы может быть исследование способов локализации проблем производительности на основе факторов SLOW, например, получение характеристик SLOW для частей программы, для отдельных вычислительных ресурсов (узлов, ядер), для заданных отрезков времени исполнения (получение динамики характеристик SLOW), определение «проблемных» фрагментов данных и вычислений и т.п.

Исследования выполнены в рамках государственного задания ИВМиМГ СО РАН FWNM-2022-0005.

Литература

1. Thoman P., Dichev K., Heller T., *et al.* A taxonomy of task-based parallel programming technologies for high-performance computing // The Journal of Supercomputing. 2018. Vol. 74. P. 1422–1434. DOI: 10.1007/s11227-018-2238-4.

2. Kaiser H., Heller T., Adelstein-Lelbach B., *et al.* HPX: A Task Based Programming Model in a Global Address Space // 8th International Conference on PGAS Programming Models, PGAS'2014, Eugene OR, USA, October 6–10, 2014. Proceedings. Article 6. ACM, 2014. P. 1–11. DOI: 10.1145/2676870.2676883.

3. Malyshkin V.E., Perepelkin V.A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // 11th International Conference on Parallel Computing Technologies, PaCT-2011, Kazan, Russia, September 19–23, 2011. Proceedings. Vol. 6873 / ed. by V. Malyshkin. Springer, 2011. P. 53–61. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-23178-0_5.

4. Shende S., Malony A.D. The TAU Parallel Performance System // International Journal of High Performance Computing Applications. 2006. Vol. 20, no. 2. P. 287–311. DOI: 10.1177/1094342006064482.

5. Lorenz D., Feld C. Scaling Score-P to the next level // Procedia Computer Science. 2017. Vol. 108. P. 2180–2189. DOI: 10.1016/j.procs.2017.05.107.

6. Extrae instrumentation package. URL: `https://tools.bsc.es/extrae` (дата обращения: 23.04.2024).

7. Vampir 10.4. URL: `https://vampir.eu/` (дата обращения: 23.04.2024).

8. Zhukov I., Feld C., Geimer M., *et al.* Scalasca v2: Back to the Future // 8th International Workshop on Parallel Tools for High Performance Computing, HLRS, Stuttgart, Germany, October, 2014. Proceedings. Ed. by C. Niethammer, J. Gracia, A. Knüpfer, *et al.* Springer, 2015. P. 1–24. DOI: 10.1007/978-3-319-16012-2_1.

9. Mantovani F., Calore E. Multi-Node Advanced Performance and Power Analysis with Paraver // Advances in Parallel Computing. Vol. 32. IOS Press, 2018. P. 723–732. DOI: 10.3233/978-1-61499-843-3-723.

10. Intel Trace Analyzer and Collector. URL: `https://www.intel.com/content/www/us/en/developer/tools/oneapi/trace-analyzer.html` (дата обращения: 23.04.2024).

11. Huynh A., Thain D., Pericàs M., Taura K. DAGViz: a DAG visualization tool for analyzing task-parallel program traces // 2nd WS on Visual Performance Analysis, VPA '15, November, 2015. Proceedings. No. 3. ACM, 2015. P. 1–8. DOI: 10.1145/2835238.2835241.

12. Huynh A., Taura K. Delay Spotter: A Tool for Spotting Scheduler-Caused Delays in Task Parallel Runtime Systems // 2017 IEEE International Conference on Cluster Computing, CLUSTER, Honolulu, HI, USA, September 5–8, 2017. IEEE, 2017. P. 114–125. DOI: 10.1109/CLUSTER.2017.82.

13. Ceballos G., Grass T., Hugo A., Black-Schaffer D. Analyzing performance variation of task schedulers with TaskInsight // Parallel Computing. 2018. Vol. 75. P. 11–27. DOI: 10.1016/j.parco.2018.02.003.

14. Pinto V.G. Performance Analysis Strategies for Task-based Applications on Hybrid Platforms: PhD thesis / Vinícius Garcia Pinto. Universidade Federal do Rio Grande do Sul - UFRGS, Brazil, UGA - Université Grenoble Alpes, France, 2018. URL: `https://theses.hal.science/tel-01962333`.

15. Pinto V.G., Nesi L.L., Miletto M.C., Schnorr L.M. Providing In-depth Performance Analysis for Heterogeneous Task-based Applications with StarVZ // 2021 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, Portland, OR, USA, June 17–21, 2021. IEEE, 2021. P. 16–25. DOI: 10.1109/IPDPSW52791.2021.00013.

16. Малышкин В.Э. Технология фрагментированного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2012. № 46(305), Вып. 1. С. 45–55. DOI: 10.14529/cmse120104.

17. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability // Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

18. Dokulil J., Benkner S. The OCR-Vx experience: lessons learned from designing and implementing a task-based runtime system // The Journal of Supercomputing. 2022. Vol. 78. P. 12344–12379. DOI: 10.1007/s11227-022-04355-0.

19. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions // International Conference on High Performance Computing, Networking, Storage and Analysis, SC'12, Salt Lake City, UT, USA, November 10–16, 2012. Proceedings. IEEE, 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

20. Akhmed-Zaki D., Lebedev D., Perepelkin V. Implementation of a three dimensional three-phase fluid flow ("oil-water-gas") numerical model in LuNA fragmented programming system // The Journal of Supercomputing. 2017. Vol. 73. P. 624–630. DOI: 10.1007/s11227-016-1780-1.

21. Daribayev B., Perepelkin V., Lebedev D., Akhmed-Zaki D. Implementation of the Two-Dimensional Elliptic Equation Model in LuNA Fragmented Programming System // IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018, Almaty, Kazakhstan, October 17–19, 2018. Proceedings. IEEE, 2018. P. 161–164. DOI: 10.1109/ICAICT.2018.8747145.

22. Akhmed-Zaki D., Lebedev D., Perepelkin V. Implementation of a 3D model heat equation using fragmented programming technology // The Journal of Supercomputing. 2019. Vol. 75, no. 12. P. 7827–7832. DOI: 10.1007/s11227-018-2710-1.

23. Akhmed-Zaki D., Lebedev D., Malyshkin V., Perepelkin V. Automated Construction of High Performance Distributed Programs in LuNA System // 15th International Conference on Parallel Computing Technologies, PaCT 2019, Almaty, Kazakhstan, August 19–23. Proceedings. Vol. 11657 / ed. by V. Malyshkin. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.

24. Малышкин В.Э., Перепелкин В.А. Мультиагентный подход к повышению эффективности исполнения фрагментированных программ в системе LuNA // Проблемы информатики. 2023. № 3. С. 55–67. DOI: 10.24412/2073-0667-2023-3-55-67.

25. Belyaev N., Kireev S. LuNA-ICLU Compiler for Automated Generation of Iterative Fragmented Programs // 15th International Conference on Parallel Computing Technologies, PaCT 2019, Almaty, Kazakhstan, August 19–23, 2019. Proceedings. Vol. 11657 / ed. by V. Malyshkin. Springer, 2019. P. 10–17. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_2.

26. Belyaev N., Perepelkin V. High-Efficiency Specialized Support for Dense Linear Algebra Arithmetic in LuNA System // 16th International Conference on Parallel Computing Technologies, PaCT 2021, Kaliningrad, Russia, September 13–18, 2021. Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 143–150. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_11.

27. Malyshkin V., Perepelkin V. Trace-Based Optimization of Fragmented Programs Execution in LuNA System // 16th International Conference on Parallel Computing Technologies, PaCT 2021, Kaliningrad, Russia, September 13–18, 2021. Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 3–10. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_1.

28. Malyshkin V., Perepelkin V., Lyamin A. Trace Balancing Technique for Trace Playback in LuNA System // 17th International Conference on Parallel Computing Technologies, PaCT 2023, Astana, Kazakhstan, August 21–25, 2023. Proceedings. Vol. 14098 /

ed. by V. Malyshkin. Springer, 2023. P. 42–50. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-41673-6_4.

29. Межведомственный Суперкомпьютерный Центр Российской Академии Наук. URL: https://www.jscc.ru/ (дата обращения: 23.04.2024).

30. Kireev S. A Parallel 3D Code for Simulation of Self-gravitating Gas-Dust Systems // 10th International Conference on Parallel Computing Technologies, PaCT 2009, Novosibirsk, Russia, August 31 – September 4, 2009. Proceedings. Vol. 5698 / ed. by V. Malyshkin. Springer, 2009. P. 406–413. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-03275-2_40.

Киреев Сергей Евгеньевич, лаборатория синтеза параллельных программ, Институт вычислительной математики и математической геофизики СО РАН, кафедра параллельных вычислений, Новосибирский государственный университет (Новосибирск, Российская Федерация)

Литвинов Василий Сергеевич, Новосибирский государственный университет (Новосибирск, Российская Федерация)

DOI: 10.14529/cmse240205

ANALYSIS OF FRAGMENTED PROGRAMS EXECUTION BASED ON SLOW FACTORS

© 2024 S.E. Kireev[1,2], V.S. Litvinov[2]

[1] *Institute of Computational Mathematics and Mathematical Geophysics SB RAS (pr. Lavrentieva 6, Novosibirsk, 630090 Russia),*

[2] *Novosibirsk State University (str. Pirogova 2, Novosibirsk, 630090 Russia)*

E-mail: kireev@ssd.sscc.ru, v.litvinov@g.nsu.ru

Received: 10.05.2024

When executing parallel programs based on the task parallelism paradigm, several issues need to be addressed, such as choosing the order in which tasks are started, considering the dependencies between them, distributing data and tasks across parallel processes, and balancing resource utilization. These issues fall under the category of system-level parallel programming and are typically handled by a dedicated execution system. The final performance of a parallel program depends on how effectively these issues are addressed, as well as the structure and characteristics of the underlying algorithm. If the program's performance is insufficient, optimization may be required, which necessitates identifying the bottlenecks that hinder its performance. Profiling can be used to pinpoint program bottlenecks by collecting performance metrics that may reveal the source of performance issues. However, the conventional tools commonly used for profiling parallel programs are not able to provide an answer in terms of the required concepts, due to the difficulty in analyzing the asynchronous execution of multiple tasks, as well as the inability to differentiate between application (multiple tasks) and system (operating system) components within an executing program. Consequently, such programs necessitate the development of novel profiling and analysis techniques. The paper discusses the problem of obtaining comprehensible performance characteristics of task-based parallel programs for performance analysis and optimization. It is suggested to evaluate the influence of the following factors: Starvation, Latency, Overhead and Waiting for contention resolution (SLOW). An algorithm for obtaining the corresponding characteristics for the LuNA fragmented programming system is presented, as well as a method for analyzing them to optimize LuNA programs. The correctness of the approach has been demonstrated on a number of synthetic tests. The application of the approach to the analysis of the "real-world" numerical simulation program is shown.

Keywords: performance analysis, parallel programming, fragmented programming, task parallelism, LuNA system.

FOR CITATION

Kireev S.E., Litvinov V.S. Analysis of the Execution of Fragmented Programs Based on SLOW Factors. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 77–96. (in Russian) DOI: 10.14529/cmse240205.

References

1. Thoman P., Dichev K., Heller T., *et al.* A taxonomy of task-based parallel programming technologies for high-performance computing. The Journal of Supercomputing. 2018. Vol. 74. P. 1422–1434. DOI: 10.1007/s11227-018-2238-4.
2. Kaiser H., Heller T., Adelstein-Lelbach B., *et al.* HPX: A Task Based Programming Model in a Global Address Space. 8th International Conference on PGAS Programming Models, PGAS'2014, Eugene OR, USA, October 6–10, 2014. Proceedings. Article 6. ACM, 2014. P. 1–11. DOI: 10.1145/2676870.2676883.
3. Malyshkin V.E., Perepelkin V.A. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem. 11th International Conference on Parallel Computing Technologies, PaCT-2011, Kazan, Russia, September 19–23, 2011. Proceedings. Vol. 6873 / ed. by V. Malyshkin. Springer, 2011. P. 53–61. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-23178-0_5.
4. Shende S., Malony A.D. The TAU Parallel Performance System. International Journal of High Performance Computing Applications. 2006. Vol. 20, no. 2. P. 287–311. DOI: 10.1177/1094342006064482.
5. Lorenz D., Feld C. Scaling Score-P to the next level. Procedia Computer Science. 2017. Vol. 108. P. 2180–2189. DOI: 10.1016/j.procs.2017.05.107.
6. Extrae instrumentation package. URL: `https://tools.bsc.es/extrae` (accessed: 23.04.2024).
7. Vampir 10.4. URL: `https://vampir.eu/` (accessed: 23.04.2024).
8. Zhukov I., Feld C., Geimer M., *et al.* Scalasca v2: Back to the Future. 8th International Workshop on Parallel Tools for High Performance Computing, HLRS, Stuttgart, Germany, October, 2014. Proceedings. Ed. by C. Niethammer, J. Gracia, A. Knüpfer, *et al.* Springer, 2015. P. 1–24. DOI: 10.1007/978-3-319-16012-2_1.
9. Mantovani F., Calore E. Multi-Node Advanced Performance and Power Analysis with Paraver. Advances in Parallel Computing. Vol. 32. IOS Press, 2018. P. 723–732. DOI: 10.3233/978-1-61499-843-3-723.
10. Intel Trace Analyzer and Collector. URL: `https://www.intel.com/content/www/us/en/developer/tools/oneapi/trace-analyzer.html` (accessed: 23.04.2024).
11. Huynh A., Thain D., Pericàs M., Taura K. DAGViz: a DAG visualization tool for analyzing task-parallel program traces. 2nd WS on Visual Performance Analysis, VPA '15, November, 2015. Proceedings. No. 3. ACM, 2015. P. 1–8. DOI: 10.1145/2835238.2835241.

12. Huynh A., Taura K. Delay Spotter: A Tool for Spotting Scheduler-Caused Delays in Task Parallel Runtime Systems. 2017 IEEE International Conference on Cluster Computing, CLUSTER, Honolulu, HI, USA, September 5–8, 2017. IEEE, 2017. P. 114–125. DOI: 10.1109/CLUSTER.2017.82.

13. Ceballos G., Grass T., Hugo A., Black-Schaffer D. Analyzing performance variation of task schedulers with TaskInsight. Parallel Computing. 2018. Vol. 75. P. 11–27. DOI: 10.1016/j.parco.2018.02.003.

14. Pinto V.G. Performance Analysis Strategies for Task-based Applications on Hybrid Platforms: PhD thesis / Vinícius Garcia Pinto. Universidade Federal do Rio Grande do Sul - UFRGS, Brazil, UGA - Université Grenoble Alpes, France, 2018. URL: `https://theses.hal.science/tel-01962333`.

15. Pinto V.G., Nesi L.L., Miletto M.C., Schnorr L.M. Providing In-depth Performance Analysis for Heterogeneous Task-based Applications with StarVZ. 2021 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW, Portland, OR, USA, June 17–21, 2021. IEEE, 2021. P. 16–25. DOI: 10.1109/IPDPSW52791.2021.00013.

16. Malyshkin V.E. Fragmented programming technology. Bulletin of the South Ural State University. Computational Mathematics and Software Engineering. 2012. No. 46(305), Iss. 1. P. 45–55. (in Russian) DOI: 10.14529/cmse120104.

17. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability. Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

18. Dokulil J., Benkner S. The OCR-Vx experience: lessons learned from designing and implementing a task-based runtime system. The Journal of Supercomputing. 2022. Vol. 78. P. 12344–12379. DOI: 10.1007/s11227-022-04355-0.

19. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions. International Conference on High Performance Computing, Networking, Storage and Analysis, SC'12, Salt Lake City, UT, USA, November 10–16, 2012. Proceedings. IEEE, 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

20. Akhmed-Zaki D., Lebedev D., Perepelkin V. Implementation of a three dimensional three-phase fluid flow ("oil-water-gas") numerical model in LuNA fragmented programming system. The Journal of Supercomputing. 2017. Vol. 73. P. 624–630. DOI: 10.1007/s11227-016-1780-1.

21. Daribayev B., Perepelkin V., Lebedev D., Akhmed-Zaki D. Implementation of the Two-Dimensional Elliptic Equation Model in LuNA Fragmented Programming System. IEEE 12th International Conference on Application of Information and Communication Technologies, AICT 2018, Almaty, Kazakhstan, October 17–19, 2018. Proceedings. IEEE, 2018. P. 161–164. DOI: 10.1109/ICAICT.2018.8747145.

22. Akhmed-Zaki D., Lebedev D., Perepelkin V. Implementation of a 3D model heat equation using fragmented programming technology. The Journal of Supercomputing. 2019. Vol. 75, no. 12. P. 7827–7832. DOI: 10.1007/s11227-018-2710-1.

23. Akhmed-Zaki D., Lebedev D., Malyshkin V., Perepelkin V. Automated Construction of High Performance Distributed Programs in LuNA System. 15th International Conference on Parallel Computing Technologies, PaCT 2019, Almaty, Kazakhstan, August 19–23, 2019.

Proceedings. Vol. 11657 / ed. by V. Malyshkin. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.

24. Malyshkin V.E., Perepelkin V.A. A Multi-Agent Approach to Improve Execution Efficiency of Fragmented Programs in LuNA System. Problemy Informatiki. 2023. No. 3. P. 55–67. (in Russian) DOI: 10.24412/2073-0667-2023-3-55-67.

25. Belyaev N., Kireev S. LuNA-ICLU Compiler for Automated Generation of Iterative Fragmented Programs. 15th International Conference on Parallel Computing Technologies, PaCT 2019, Almaty, Kazakhstan, August 19–23, 2019. Proceedings. Vol. 11657 / ed. by V. Malyshkin. Springer, 2019. P. 10–17. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_2.

26. Belyaev N., Perepelkin V. High-Efficiency Specialized Support for Dense Linear Algebra Arithmetic in LuNA System. 16th International Conference on Parallel Computing Technologies, PaCT 2021, Kaliningrad, Russia, September 13–18, 2021. Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 143–150. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_11.

27. Malyshkin V., Perepelkin V. Trace-Based Optimization of Fragmented Programs Execution in LuNA System. 16th International Conference on Parallel Computing Technologies, PaCT 2021, Kaliningrad, Russia, September 13–18, 2021. Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 3–10. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_1.

28. Malyshkin V., Perepelkin V., Lyamin A. Trace Balancing Technique for Trace Playback in LuNA System. 17th International Conference on Parallel Computing Technologies, PaCT 2023, Astana, Kazakhstan, August 21–25, 2023. Proceedings. Vol. 14098 / ed. by V. Malyshkin. Springer, 2023. P. 42–50. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-41673-6_4.

29. Joint SuperComputer Center of the Russian Academy of Sciences. URL: `https://www.jscc.ru/` (accessed: 23.04.2024) (in Russian).

30. Kireev S. A Parallel 3D Code for Simulation of Self-gravitating Gas-Dust Systems. 10th International Conference on Parallel Computing Technologies, PaCT 2009, Novosibirsk, Russia, August 31 – September 4, 2009. Proceedings. Vol. 5698 / ed. by V. Malyshkin. Springer, 2009. P. 406–413. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-03275-2_40.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `https://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
2024 Том 13, № 2

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 31.05.2024. Дата выхода в свет 28.06.2024. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 11,62. Тираж 500 экз. Заказ 110/214. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.