DOI 10.31891/2307-5732-2025-347-32
УДК 004.51

ЛУК'ЯНЕЦЬ МИХАЙЛО
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
https://orcid.org/0009-0006-0083-391X
e-mail: mihail.lukjanec@gmail.com
СУЛЕМА ЄВГЕНІЯ
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
https://orcid.org/0000-0001-7871-9806
e-mail: sulema@pzks.fpm.kpi.ua

МЕТОД ВІДОБРАЖЕННЯ ТОЧКОВИХ ТЕМПОРАЛЬНИХ МУЛЬТИМОДАЛЬНИХ ПОТОКОВИХ ДАНИХ ПРИСТРОЇВ IOT ЗІ ЗБЕРЕЖЕННЯМ СТАБІЛЬНОСТІ ВІДОБРАЖЕННЯ ПОПЕРЕДНЬО ВІЗУАЛІЗОВАНИХ ДАНИХ

Стаття присвячена розробленню методу відображення темпоральних мультимодальних потокових даних пристроїв IoT із забезпеченням стабільності раніше візуалізованих даних. Актуальність пояснюється необхідністю візуалізації великих обсягів даних IoT у реальному часі, що важливо для роботи з даними. Мета дослідження полягає у зменшенні навантаження на систему без втрати контексту даних. Запропоновано метод, що використовує даунсемплінг із збереженням стабільності відображення даних шляхом обмеження часового проміжку ресемплінгу. У статті продемонстровано результати застосування методу на основі запропонованої імплементації.

Ключові слова: інженерія програмного забезпечення, візуалізація, потокові дані, Інтернет речей.

MYKHAILO LUKIANETS, YEVGENIYA SULEMA
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"

METHOD FOR VISUALIZING POINT-BASED TEMPORAL MULTIMODAL STREAMING DATA FROM IOT DEVICES WHILE MAINTAINING THE STABILITY OF PREVIOUSLY VISUALIZED DATA

This paper presents a method for visualizing temporal multimodal streaming data from Internet of Things (IoT) devices, ensuring the stability of previously visualized data. The growing volume of real-time IoT data presents challenges in visualization and analysis, requiring efficient techniques for handling large datasets in dynamic environments. The goal of this research is to reduce system load without losing the context of the data being visualized.

The proposed method uses downsampling techniques to reduce the volume of displayed data while maintaining visualization stability by limiting the time window for data resampling. This method optimizes system resources, ensuring that users can interact with real-time data without cognitive overload, even when processing large volumes of data continuously. Moreover, the research shows that the proposed method can handle the challenges of real-time data streams effectively, providing a better user experience by preventing cognitive overload. Users are presented with only the most relevant data, ensuring that the visualization remains clear, stable, and easy to interpret.

The paper demonstrates the effectiveness of the method through an implementation using the SciChart library for .NET, showcasing its ability to reduce system resource consumption and improve real-time data representation. The results reveal that the proposed approach significantly reduces CPU and GPU load, making it suitable for real-time IoT data visualization, especially in resource-constrained environments.

In conclusion, this method offers a scalable and efficient solution for visualizing temporal multimodal IoT data, balancing the need to reduce system load while maintaining stable, clear, and meaningful visualizations. This approach ensures that real-time data remains accessible and interpretable, even when large amounts of data are continuously processed.

Keywords: software engineering, *visualization, point data, Internet of Things.*

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Мережі пристроїв Інтернету речей (IoT) стають все більш розповсюджені у теперішній час. У сучасних системах Інтернету речей (IoT) генерується величезна кількість даних, які мають темпоральний і мультимодальний характер. Такі дані надходять у потоковому режимі від різноманітних сенсорів і пристроїв, що викликає потребу в ефективних методах їх візуалізації для прийняття рішень у реальному часі.

Проблема ускладнюється необхідністю збереження стабільності відображення раніше візуалізованих даних при оновленні інформації. Темпоральний характер даних, що постійно надходять, може спричинити зміну попередньо відображених даних, призводить до втрати контексту та труднощів в аналізі. Це підкреслює актуальність розроблення нового методу візуалізації, який би дозволяв ефективно обробляти й відображати потокові даних із мінімізацією когнітивного навантаження на користувача внаслідок зміни відображення.

Таким чином, існує необхідність у розробленні ефективного методу відображення точкових темпоральних мультимодальних потокових даних пристроїв IoT, який би забезпечував незмінність попередньо відображених даних, при цьому з найменшим можливим навантаженням на ресурси ПК.

Аналіз досліджень та публікацій

Широко відомою проблемою в обробці даних є задача зміни їх кількості з метою адаптації до різних завдань обчислення та візуалізації — ресемплінг [1-2]. Ця задача є актуальною в багатьох сферах, зокрема при обробці потокових даних, аналізі часових рядів, машинному навчанні та роботі з великими

обсягами інформації. Одним із ключових аспектів ресемплінгу є даунсемплінг — процес зменшення кількості даних, що дозволяє знизити обчислювальні витрати, зберігаючи при цьому основну структуру вихідної інформації [3, 4].

На сьогодні відомо багато методів ресемплінгу точкових даних, які знаходять застосування у різних контекстах. Значна частина цих методів стосується обробки хмар точок — множин точкових даних, що зазвичай відображаються у тривимірному просторі. Такі методи особливо важливі для задач комп'ютерної графіки, 3D-моделювання, обробки геопросторових даних і комп'ютерного зору, де потрібна оптимізація великої кількості точок при збереженні форми об'єкта [5-7].

Крім того, існують ефективні підходи до ресемплінгу двовимірних даних, які використовуються для задач візуалізації на площині, таких як побудова графіків і карт, обробка зображень та аналіз двовимірних часових рядів [3]. Ресемплінг на площині дозволяє адаптувати дані для їх подальшого аналізу, знижуючи обсяг інформації, але зберігаючи основні характеристики, необхідні для інтерпретації.

Окремо слід зазначити, що задача ресемплінгу потокових даних зі збереженням стабільності є важливою. У контексті систем реального часу, таких як мережі IoT, де дані надходять безперервно, важливо зберігати стабільність відображення вже візуалізованих даних. Це дозволяє уникнути постійних змін у графічному поданні, які можуть призводити до плутанини та ускладнювати аналіз. Такий підхід забезпечує більш послідовне та зрозуміле представлення даних, що особливо важливо для прийняття рішень на основі візуалізації. Збереження стабільності при ресемплінгу залишається актуальною проблемою, яка потребує розробки нових підходів та методів для її ефективного вирішення.

Формулювання цілей статті

Метою роботи є зменшення навантаження на комп’ютерну систему без втрати контексту даних під час відображення темпоральних мультимодальних потокових даних пристроїв IoT та забезпечення стабільності візуалізації раніше відображених даних при оновленні відображуваної інформації.

Виклад основного матеріалу

З розвитком технологій Інтернету речей (IoT) обсяг даних, що генерується різноманітними пристроями, невпинно зростає. Ці дані є критично важливими для забезпечення ефективної роботи систем у реальному часі, оскільки вони дозволяють здійснювати моніторинг, прогнозування, аналіз поведінки об’єктів, виявлення аномалій і підтримку прийняття рішень. Важливість візуалізації цих даних не можна переоцінити, адже графічне представлення дозволяє користувачам швидко оцінювати ситуацію, виявляти тренди та аналізувати взаємозв’язки між різними параметрами.

Одним із ефективних підходів для візуалізації потокових IoT-даних є їх відображення у вигляді точкових графіків на 2D або 3D площині [8]. Цей метод надає можливість відобразити темпоральні аспекти даних разом із просторовими або іншими атрибутами, що значно спрощує аналіз. Наприклад, 2D-візуалізація може бути використана для моніторингу залежності між часом і певним параметром, тоді як 3D-відображення додає ще один вимір, що дозволяє представляти складніші зв’язки між кількома змінними. Приклад такого 2D-відображення наведений на рис.1. Використання анімації або кольорових маркерів може додатково акцентувати увагу на змінах у реальному часі.

Проте при зростанні кількості пристроїв і частоти збору даних такі методи візуалізації стикаються зі значними викликами. Обмеження апаратних ресурсів, а також когнітивні можливості користувачів, ускладнюють роботу із занадто великою кількістю інформації. Постійне оновлення даних у реальному часі може призводити до перевантаження інтерфейсу візуалізації, що ускладнює розуміння загальної картини і заважає оперативному прийняттю рішень.

Для зменшення навантаження часто застосовують методи даунсемплінгу, які передбачають зниження обсягу даних для відображення. Даунсемплінг дозволяє вибірково зберігати й візуалізувати лише частину даних, відсікаючи надмірну кількість точок. Це допомагає зменшити складність візуальної інформації і підвищити продуктивність системи. Наприклад, замість відображення всіх зібраних точок даних за певний період, система може показувати лише репрезентативні точки, що дозволяє користувачу зосередитися на ключових аспектах.

Однак використання даунсемплінгу породжує нову проблему — зміна або видалення раніше візуалізованих даних. Це може спричинити втрату контексту та ускладнити розуміння динаміки змін, особливо коли користувач потребує порівняння поточних даних із історичними. Змінюваність візуалізації викликає плутанину, що є серйозною перешкодою для аналітичної роботи.

Вирішення цієї проблеми потребує розробки методу, який забезпечував би стабільність відображення раніше візуалізованих даних, навіть при оновленні або застосуванні даунсемплінгу. Це дозволить зберігати контекст і надавати користувачам незмінну візуальну основу, на якій можуть відображатися нові дані без перекручування попередніх візуалізованих результатів. Такий підхід сприятиме зниженню когнітивного навантаження на користувачів, дозволяючи їм ефективно працювати навіть із великими потоками інформації, що є критично важливим для багатьох застосувань IoT.

Відповідно запропоновано метод, алгоритм реалізації якого зображений на рис.2. Робота починається з ініціалізації параметрів. На цьому етапі встановлюються базові налаштування роботи системи, такі як часовий інтервал оброблення даних, параметри даунсемплінгу (зокрема, частота вибірки, допустимі межі зменшення кількості даних або коефіцієнт зменшення даних), а також межі

групування даних за їхніми параметрами. Крім того, налаштовуються параметри графічного відображення, зокрема тип графіка (2D або 3D), кольорові схеми та стиль маркерів.

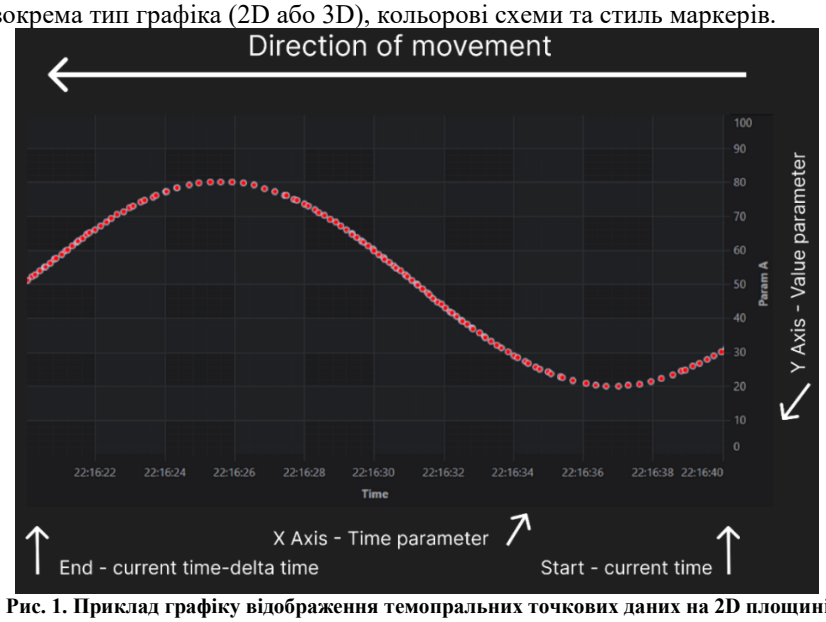

Рис. 1. Приклад графіку відображення темопральних точкових даних на 2D площині

Далі система переходить у стан очікування надходження даних. Постійно очікується оновлення від потоків даних, що надходять від IoT пристроїв, і відповідно відбувається реагування на появу нових даних. Після отримання сигналу про нові дані виконується їх перевірка, яка включає визначення наявності даних та їхньої відповідності заданим параметрам (наприклад, чи дані належать до необхідного часового інтервалу).

Після цього дані, що надійшли, розподіляються на три групи. Перша група — це застарілі дані, які виходять за межі визначеного часовим інтервалом вікна. Вони більше не мають аналітичної цінності та не беруть участі в ресемплінгу, а лише зберігаються для архіву або відразу видаляються для звільнення ресурсів. Серед усіх наявних даних $A = \langle a_i \rangle_{i=1}^{n}$ вони виділяються за формулою (1), де $t_{current}$ є часом початку обрахунку, $t_{display}$ є максимальним часом відображення даних, а t_i є часом надходження даних a_i. Друга група — це стабільні дані, які вже були відображені раніше і не можуть бути змінені для збереження візуальної стабільності. Такі дані визначаються за формулою (2), у якій $t_{resample}$ є максимально дозволеним часом для ресемплінгу даних. Третя група складається з нових змінних даних, які ще не зафіксовані у візуалізації та можуть бути піддані ресемплінгу, визначається за формулою (3).

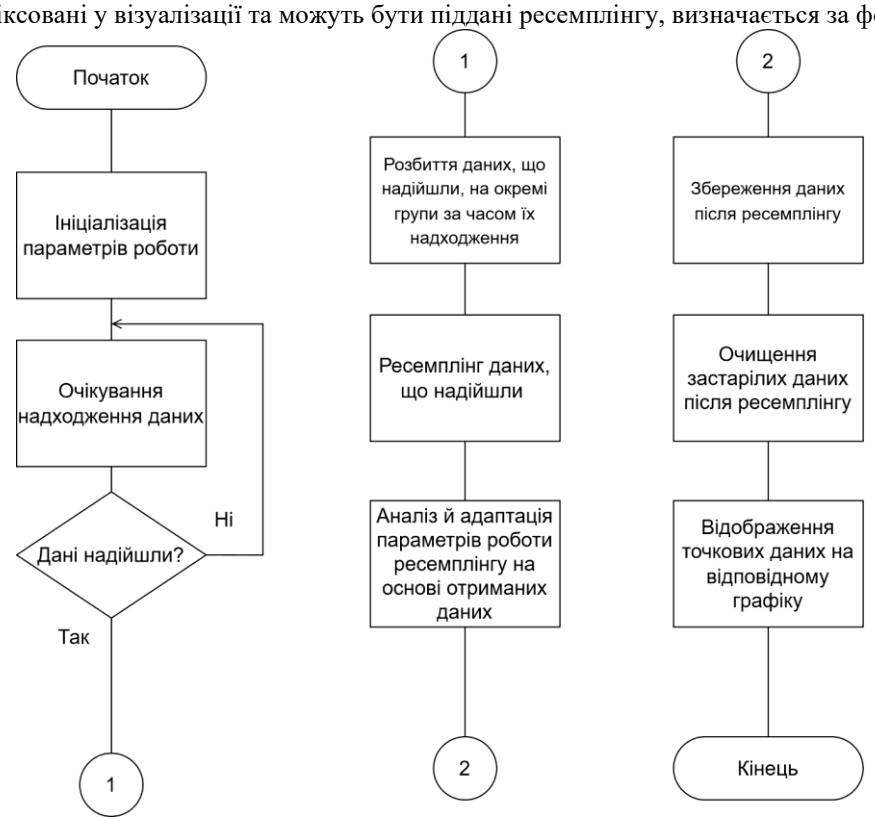

Рис. 2. Блок-схема алгоритму відображення потокових даних з їх ресемплінгом та зберіганням візуалізації попередньо відображених даних

$$d_i = \begin{cases} a_i, & t_i < t_{current} - t_{display}, \\ \emptyset, & t_i \geq t_{current} - t_{display}. \end{cases} \quad (1)$$

$$c_i = \begin{cases} a_i, & t_{current} - t_{display} \leq t_i < t_{current} - t_{resample}, \\ \emptyset, & \text{інакше.} \end{cases} \quad (2)$$

$$r_i = \begin{cases} a_i, & t_{current} - t_{resample} \leq t_i < t_{current}, \\ \emptyset, & \text{інакше.} \end{cases} \quad (3)$$

Ресемплінг виконується виключно над змінною групою даних $R = \langle r_i \rangle_{i=1}^{n}$. На цьому етапі застосовуються алгоритми зниження кількості точок даних, наприклад, через вибір середніх значень, екстремальних точок або рівномірне зменшення вибірки. Це дозволяє зменшити обсяг інформації, що підлягає візуалізації, зберігаючи при цьому основну структуру і тренди даних. Після завершення ресемплінгу виконуються аналіз і адаптація параметрів. Зокрема, оцінюється ефективність зниження навантаження, аналізуються динамічні зміни в потоках даних і, за потреби, коригуються параметри ресемплінгу, щоб оптимізувати баланс між деталізацією і продуктивністю.

Після ресемплінгу дані зберігаються у внутрішніх структурах пам'яті системи. Стабільні дані $C = \langle c_i \rangle_{i=1}^{n}$ залишаються без змін і використовуються для подальшого відображення, тоді як змінні дані оновлюють графічний буфер. Очищення застарілих даних $D = \langle d_i \rangle_{i=1}^{n}$ проводиться, щоб звільнити ресурси та забезпечити зберігання лише актуальної інформації. На фінальному етапі виконується відображення оброблених даних на графіку. Відображення поєднує стабільні та змінні дані, забезпечуючи користувачів актуальною і зрозумілою візуальною інформацією. Графік автоматично оновлюється, зберігаючи стабільність попередньо відображених даних і додаючи нові змінні, що дозволяє в реальному часі відслідковувати динаміку потокових IoT даних.

Варіація методу також передбачає можливість паралельного видалення застарілих даних, що є важливим для підтримання актуальності візуалізації навіть у тих випадках, коли нові дані тимчасово не надходять. Цей механізм дозволяє системі динамічно очищати старі дані, які виходять за межі визначеного часовим інтервалом $t_{display}$, забезпечуючи тим самим збереження ресурсів і запобігання переповненню пам'яті.

Паралельне видалення застарілих даних здійснюється незалежно від основного потоку обробки нових даних. Це означає, що навіть за відсутності нових вхідних даних алгоритм продовжує перевіряти поточний стан усіх збережених точок і автоматично видаляє ті, що більше не мають аналітичної цінності. Такий підхід забезпечує підтримання актуальної візуалізації, де на графіку завжди відображаються тільки найсвіжіші та релевантні дані, що входять до $t_{display}$.

Ця можливість має важливе значення для систем, що працюють у режимі реального часу, де навіть короткочасна затримка надходження нових даних не повинна впливати на якість візуалізації. Завдяки паралельному видаленню старих даних користувач завжди бачить лише релевантну інформацію, що дозволяє зберігати цілісність і стабільність графічного представлення. Крім того, цей підхід сприяє оптимізації використання ресурсів системи, зменшуючи обсяг даних, які необхідно зберігати та обробляти.

Для перевірки роботи запропонованого методу була створена реалізація з використанням бібліотеки SciChart для .NET [9, 10]. Завдяки її швидкодії можливо порівнювати роботу методу на різноманітній, в тому числі і великій кількості даних на хвилину і також можливо створити застосунок, що відображав би дані як в 3D, так і 2D площині, що було необхідно для порівняння навантаження в різних режимах роботи.

Для порівняння роботи методу також була створена реалізація, що робила би аналогічне відображення даних, проте без проведення даунсемплінгу даних, при цьому зберігаючи постійне додавання нових даних на видалення застарілих. На рис. 3 та рис. 4 зображені результати порівняння. Екземпляри застосунку з увімкненими та вимкненим даунсемпліногом, різною кількістю даних за хвилину, що надходять, та різним відображенням (2D або 3D) були запущені на даній конфігурації: win10-x64, .NET 8, intel i7-7700HQ, GeForce GTX 1050 4GB, RAM 16GB.

На рис. 3 відображені власне дані навантаження на CPU. З отриманих результатів можна зробити висновок, що при застосуванні ресемплінгу навантаження було меншим завдяки меншій кількості даних, що необхідно було відобразити. Також варто зауважити, що в навантаження на CPU також входить і обчислення власне ресемплінгу даних, а відповідно навантаження від його використання значно менше ніж звільнені ресурси від зменшення кількості відображуваних даних.

На рис. 4 зображені аналогічні дані, проте заміри навантаження були проведені для GPU. З наведених даних можна зробити висновок, що навантаження на GPU з використанням ресемплінгу є майже сталим, оскільки кількість даних, що відображається, є стабільною завдяки проведеному ресемплінгу.

Варто зауважити, що навантаження при відображенні даних на 2D площині є більшим, ніж на 3D, що ймовірно пояснюється особливостями реалізації бібліотеки, а саме тим, що для відображення на 3D використовувалися інструменти DirectX, що краще оптимізовані для відображення різноманітних елементів 3D графіки. Оскільки наведена сцена з певною кількістю точок є досить проста відносно сцен, для яких зазвичай використовується 3D відображення, то навантаження було нижчим ніж реалізоване у 2D площині на елементах WPF. Відповідно, можливо зробити висновок, що для деяких окремих випадків варто розглядати реалізацію відображення даних інструментами для 3D графіки, навіть якщо необхідно

відобразити 2D дані. Однак варто зважати на те, що в такому разі реалізація інших елементів відображення може бути ускладнена.

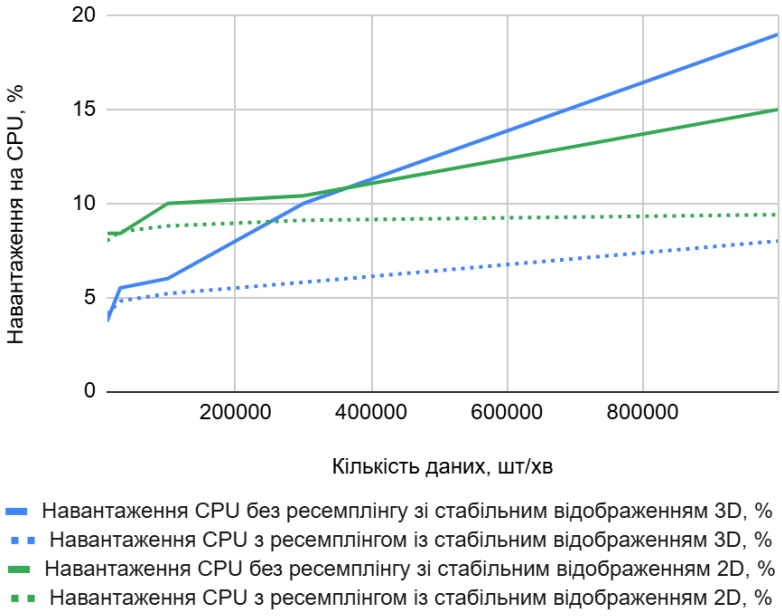

Рис. 3. Результати перевірки роботи реалізації для CPU

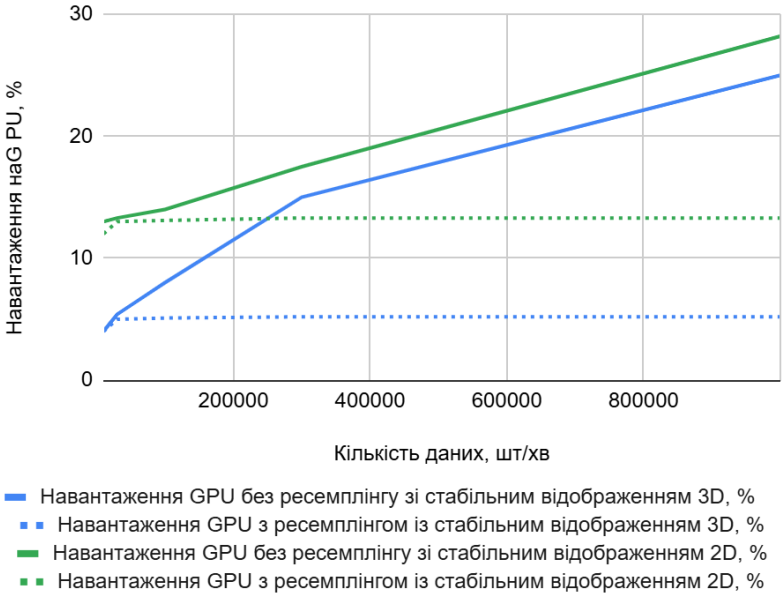

Рис. 4. Результати перевірки роботи реалізації для GPU

Загалом, розроблена реалізація показала, що застосування запропонованого методу може дійсно знизити навантаження на ресурси ПК. Серед недоліків варто зазначити необхідність встановлення початкових параметрів для роботи методу. Завеликі або замалі параметри можуть призвести до занадто малої вихідної кількості даних або ж навпаки, занадто великої. Також, одним із недоліків даного методу, є особливість його роботи у таких крайніх випадках, коли точкові дані досить рівномірно розподілені по усьому простору і, відповідно, операція ресемплінгу для них немає сенсу. При цьому ж, власне процедура ресемплінгу проводиться, що вимагає ресурсів ПК. Але, якщо дані достатньо віддалені один від одного, то вимушене додаткове навантаження не має бути значним через занадто малу кількість таких даних. Однак, розроблення алгоритму визначення необхідності застосування ресемплінгу для певного набору точкових даних є можливим вирішенням даного недоліку, оскільки можна буде оцінити доцільність використання методу і застосовувати його лише у необхідні моменти.

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі

У статті запропоновано метод відображення точкових темпоральних мультимодальних потокових даних пристроїв IoT, який включає даунсемплінг для зменшення навантаження на систему при відображенні великих обсягів даних. Використання цього методу дозволяє забезпечити стабільність візуалізації при оновленні інформації, зберігаючи важливі характеристики даних та знижуючи

когнітивне навантаження на користувачів. Метод передбачає кілька етапів: ініціалізацію параметрів системи, перевірку нових даних, їх групування, ресемплінг, очищення застарілих даних та стабільне відображення.

Результати тестування показали, що застосування даунсемплінгу дозволяє значно знизити навантаження на ресурси системи, зокрема на процесор та графічний процесор. Це досягається завдяки зменшенню кількості точок, що підлягають візуалізації.

Однак існують певні обмеження: необхідність точного налаштування параметрів даунсемплінгу для забезпечення оптимального балансу між деталізацією та продуктивністю, а також можливі труднощі при роботі з рівномірно розподіленими даними, коли застосування ресемплінгу може бути неефективним. Для вирішення цих проблем запропоновано розробити алгоритм, що оцінює доцільність використання ресемплінгу для конкретних наборів даних.

Загалом, розроблений метод показав свою ефективність в умовах реального часу і може бути використаний для візуалізації великих потоків даних в IoT системах, де критично важлива як швидкість оброблення даних, так і зручність їх інтерпретації.

Подяка

Висловлюється подяка SciChart Ltd. за надання некомерційної ліцензії для графіків у WPF та WPF 3D.

Література

1. Wang, Y., et al. (2018). A novel slicing-based regularization method for raw point clouds in visible IoT. *IEEE Access, 6*, 18299–18309. https://doi.org/10.1109/ACCESS.2018.2828873

2. James, G., et al. (2023). Resampling methods. In *An Introduction to Statistical Learning: With Applications in Python* (pp. 201–228). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-84837-0_7

3. SciChart. (n.d.). *ResamplingMode.* SciChart WPF Charting Documentation. Retrieved November 15, 2024, from https://www.scichart.com/documentation/win/current/SciChart.Data~SciChart.Data.Numerics.ResamplingMode.html

4. Eng, F., & Gustafsson, F. (2007). Downsampling non-uniformly sampled data. *EURASIP Journal on Advances in Signal Processing, 2008*(1), 1–10. https://doi.org/10.1155/2008/175629

5. Chen, S., et al. (2017). Fast resampling of three-dimensional point clouds via graphs. *IEEE Transactions on Signal Processing, 66*(3), 666–681. https://doi.org/10.1109/TSP.2017.2768441

6. Zhao, Y., Zhang, X., & Huang, X. (2022). A divide-and-merge point cloud clustering algorithm for lidar panoptic segmentation. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 7029–7035). IEEE. https://doi.org/10.1109/ICRA2022.9812321

7. Xu, Y., et al. (2022). FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking. *Neurocomputing, 494*, 255–268. https://doi.org/10.1016/j.neucom.2022.04.051

8. Palaniswami, M., et al. (2020). The role of visual assessment of clusters for big data analysis: From real-world Internet of Things. *IEEE Systems, Man, and Cybernetics Magazine, 6*(4), 45–53. https://doi.org/10.1109/MSMC.2020.3022734

9. SciChart. (n.d.). *WPF 2D Chart Library.* Retrieved November 14, 2024, from https://www.scichart.com/wpf-chart-features/

10. SciChart. (n.d.). *WPF 3D Chart Library.* Retrieved November 15, 2024, from https://www.scichart.com/wpf-3d-chart-features

References

1. Wang, Y., et al. (2018). A novel slicing-based regularization method for raw point clouds in visible IoT. IEEE Access, 6, 18299–18309. https://doi.org/10.1109/ACCESS.2018.2828873

2. James, G., et al. (2023). Resampling methods. In An Introduction to Statistical Learning: With Applications in Python (pp. 201–228). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-84837-0_7

3. SciChart. (n.d.). ResamplingMode. SciChart WPF Charting Documentation. Retrieved November 15, 2024, from https://www.scichart.com/documentation/win/current/SciChart.Data~SciChart.Data.Numerics.ResamplingMode.html

4. Eng, F., & Gustafsson, F. (2007). Downsampling non-uniformly sampled data. EURASIP Journal on Advances in Signal Processing, 2008(1), 1–10. https://doi.org/10.1155/2008/175629

5. Chen, S., et al. (2017). Fast resampling of three-dimensional point clouds via graphs. IEEE Transactions on Signal Processing, 66(3), 666–681. https://doi.org/10.1109/TSP.2017.2768441

6. Zhao, Y., Zhang, X., & Huang, X. (2022). A divide-and-merge point cloud clustering algorithm for lidar panoptic segmentation. In 2022 International Conference on Robotics and Automation (ICRA) (pp. 7029–7035). IEEE. https://doi.org/10.1109/ICRA2022.9812321

7. Xu, Y., et al. (2022). FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking. Neurocomputing, 494, 255–268. https://doi.org/10.1016/j.neucom.2022.04.051

8. Palaniswami, M., et al. (2020). The role of visual assessment of clusters for big data analysis: From real-world Internet of Things. IEEE Systems, Man, and Cybernetics Magazine, 6(4), 45–53. https://doi.org/10.1109/MSMC.2020.3022734

9. SciChart. (n.d.). WPF 2D Chart Library. Retrieved November 14, 2024, from https://www.scichart.com/wpf-chart-features/

10. SciChart. (n.d.). WPF 3D Chart Library. Retrieved November 15, 2024, from https://www.scichart.com/wpf-3d-chart-features