DOI 10.31891/2307-5732-2024-343-6-41
УДК 637.5.02

ЗАВАЛІЙ ТАРАС
Національний університет "Львівська політехніка"
https://orcid.org/0009-0002-7544-782X
e-mail: taras.i.zavalii@lpnu.ua

МОДЕЛЮВАННЯ РУХУ ПІШОХОДА НА ОСНОВІ ДАНИХ СЕНСОРІВ СМАРТФОНУ

Задача навігації завжди була актуальна в різних сферах діяльності людини. А поява дешевих електромеханічних сенсорів (IMU) спричинила нові прикладні дослідження у напрямку інерційної навігації. У статті наведено розв'язок задачі навігації пішохода з використанням смартфону та за наявності перешкод у сигналі GNSS. Дані прискорення смартфону використано для навчання LSTM моделі, яка може усувати дрейф швидкості при інтегруванні. Показано, що попри достатню точність прогнозування швидкості (MAE=0.087 м/с і R^2=0.83), кінцева обчислена траєкторія залежить від точності орієнтації в горизонтальній площині, отриманої з магнітометра.

Ключові слова: інерційна навігація, опрацювання сигналів, IMU, LSTM.

ZAVALIY TARAS
Lviv Polytechnic National University

MODELLING PEDESTRIAN MOVEMENT USING SMARTPHONE SENSOR DATA

The navigation problem has always been relevant in various fields of human activity, and the advent of cheap electromechanical sensors (IMUs) has led to new applied research in the field of inertial navigation. The main task of inertial navigation is to estimate the position of a moving object at each point in time given its initial speed and position, its acceleration and attitude. Various sensors and methods could be used for the task, including sensor fusion, step counting, visual odometry, machine learning.

The article presents a solution to the task of pedestrian navigation in urban environment. In such settings, satellite signals can suffer from dropouts and multipath effects when GNSS signals reflect off surrounding surfaces, causing the receiver to process both direct and reflected signals. Relying only on IMU sensor data to calculate distance travelled is nontrivial as the result heavily depends on the hardware characteristics and noise. That is why the main goal of the research was to solve the velocity drift problem when integrating raw acceleration in inertial navigation process.

To achieve the goal, we focused on the data preprocessing and its effect on the result. The data from the IMU sensors were sampled at high frequency, 53 Hz, resulting in a constant stream of (noisy) data. We had to filter the noise applying Butterworth low-pass filter to acceleration data and Savitzky-Golay filter to orientation data. Known orientation of the device allowed us to convert measurements from the device frame of reference to the global ENU frame. Smartphone acceleration dataset containing 133000 samples and 2500 target speed measurements was used to train an LSTM model that can regress velocity out of acceleration data. We achieved MAE of 0.087 m/s and R^2 of 0.83 on the test dataset and applied the full algorithm to reconstruct the recorded rectangular trajectory of 77×15 meters. Simple complementary filter was used for merging predicted and integrated velocities, thus eliminating drift. It was shown that despite the quite accurate speed prediction, the final result depends on the orientation in horizontal plane obtained from the magnetometer.

Key words: inertial navigation, signal processing, IMU, LSTM.

Вступ

Визначення точного розташування об'єкту є критичним для різних сервісів і технологій у логістиці, автомобільній та пішохідній навігації, військових застосуваннях. Точність загальнодоступних рішень для супутникового позиціонування сягає кількох метрів, і залежить від типу приймача, погодних умов та наявності перешкод. В умовах щільної забудови точність позиціонування смартфона знижується до 5-10 метрів [1]. Звичайно, використанням складних і дорогих приймачів з технологією RTK у найкращому випадку можна зменшити похибку до кількох сантиметрів. Але існують задачі, де потрібні альтернативні і дешевші гібридні методи, – навігація в приміщенні, навігація дронів, автономних транспортних засобів та мобільних роботизованих систем.

Ця робота реалізує основну ідею інерційної навігації для пішохода з використанням сенсорів, які можна знайти в будь-якому смартфоні. До них належать: GNSS, IMU (акселерометр, гіроскоп, магнітометр), барометр, датчик близькості тощо. Використовуючи Android-додаток Sensor Logger [2], ми зібрали дані GNSS та IMU сенсорів, попередньо опрацювали, візуалізували, об'єднали та використали їх для прогнозування координат пристрою, який тримає пішохід під час руху. Експериментально було показано, як дрейф швидкості постійно накопичується, незважаючи на фільтрацію шуму в первинних даних прискорення. Ми вирішили цю проблему, навчивши модель LSTM прогнозувати швидкість у горизонтальній площині. Це дозволило компенсувати дрейф та обчислити пройдену відстань на основі прискорення. Результати навчання LSTM моделі та результати застосування алгоритму для відновлення траєкторії пішохода подано в останньому розділі.

Аналіз останніх досліджень

Існують різні підходи до локалізації рухомих об'єктів, кожен зі своїми припущеннями та обмеженнями. Для точного визначення місцезнаходження пішохода за допомогою IMU, сенсор має бути закріплений на нозі [3], [4]. Точка з нульовою швидкістю, коли нога торкається землі, використовується для корекції дрейфу швидкості та відстані, обчислених на основі прискорення. У транспортних засобах та автономних роботизованих системах орієнтація IMU сенсора фіксована і зазвичай співпадає з напрямком руху. Для досягнення точності застосовується комплексне злиття прискорення з даними одометра, лідара чи

камери [5], [6], [7]. Методи навігації в приміщеннях використовують вхідні дані радарів або фіксованих маяків для триангуляції [8], або ж залежать від наявності карти приміщення [9]. Однак останні дослідження показують, що можливо досягти більш точної інерційної навігації за допомогою машинного навчання [10], [11].

У [10] автори описують свій метод Robust IMU Double Integration (RIDI), який базується на моделюванні швидкості пішохода на основі IMU сигналів. Вони прогнозують вектор швидкості, використовуючи історію лінійних прискорень і кутових швидкостей за допомогою алгоритму Support Vector Regression (SVR). Потім обчислюється функція корекції для виправлення низькочастотних похибок у лінійному прискоренні, завдяки чому стандартне подвійне інтегрування призводить до значно меншого дрейфу. Вдалося досягти середньої похибки позиціонування менше 3% від загальної довжини траєкторії (тобто зміщення на 5 метрів після 150 метрів ходьби). У роботі [11] для вивчення моделі руху використано декілька архітектур нейронних мереж (ResNet, LSTM, TCM) та здійснено порівняння результату навігації в приміщенні з іншими методами. На різних наборах даних ResNet показала найменшу похибку, однак проблемою стало визначення точної орієнтації пристрою в горизонтальній площині.

Нарешті, є успішні приклади розширення фільтра Калмана нейронною мережею у сфері комп'ютерного зору [12]. Автори реалізували фільтр Калмана без функції переходу стану і без фіксованих матриць коваріації Q та R. Натомість вони моделюють коваріації шуму та нелінійну функцію переходу стану за допомогою трьох окремих мереж LSTM, роблячи компоненти фільтра Калмана динамічними. На наборі даних Human 3.6M у задачі розпізнавання пози людини пропонований метод LSTM-KF перевершує стандартну LSTM та інші сучасні підходи, досягаючи в середньому 14% покращення.

З огляду на наше завдання моделювання руху без використання високоякісних сенсорів, необхідно було застосувати методи машинного навчання для досягнення стабільного результату та усунення проблеми подвійного інтегрування.

Опис методу

Для кожного інтервалу часу нам потрібно обчислити зміщення пристрою знаючи його прискорення. В основі методу лежить часткова реалізація фільтру Калмана [13], адаптованого до нашої моделі руху і наявних сенсорів. Наша реалізація фільтра є слабко зв'язаною, тобто, ми отримуємо орієнтацію (yaw) та координати GNSS без їх самостійного обчислення з первинних даних. Щоб усунути дрейф від подвійного інтегрування, ми обчислюємо зважену суму (7) між інтегрованою швидкістю та швидкістю, отриманою на виході LSTM мережі.

Задамо модель руху у двовимірній системі координат стандартними рівняннями швидкості і відстані:

$$\begin{aligned} v(t) &= v(t-1) + \int_{t-1}^{t} a(t)dt\,, \\ p(t) &= p(t-1) + \int_{t-1}^{t} v(t)dt \end{aligned} \tag{1}$$

На основі цих рівнянь побудуємо матрицю переходу стану F наступним чином:

$$F = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

а поточний стан системи опишемо вектором з прискоренням a, швидкістю v і відстанню p в глобальній системі координат ENU:

$$x = \begin{bmatrix} p \\ v \\ a \end{bmatrix} \tag{3}$$

У нашому випадку зробимо декілька припущень: початкова позиція відома, початкова швидкість і прискорення дорівнюють нулю, орієнтація пристрою співпадає з напрямком руху, і нас цікавить лише горизонтальне зміщення. Сенсори IMU калібровані засобами операційної системи Android. Різниця між географічною північчю та магнітною північчю є несуттєвою через невелику пройдену відстань на маршруті.

Точне визначення місця розташування об'єкта потребує правильного перетворення вимірювань IMU у глобальну систему координат. Використаємо систему ENU (East-North-Up), оскільки декартові координати в цій системі дозволяють проводити точні обчислення, уникаючи спотворень, які можуть виникати при кутових вимірюваннях, таких як широта та довгота. Зрештою зручно, коли осі координат в ENU системі співпадають з осями координат пристрою Android, розміщеного екраном догори і спрямованого на північ.

Для кожного інтервалу часу (~19 мс при частоті 53 Hz), алгоритм ітеративно виконує кроки 2-4 на вхідних потоках попередньо опрацьованих даних:

1. Попереднє опрацювання.
   a. Відфільтрувати шум в початкових даних прискорення (Butterworth Low-pass).
   b. Відфільтрувати шум в даних орієнтації (Savitzki-Golay).
   c. Синхронізувати дані прискорення та орієнтації у часі.
   d. Перетворити вектор прискорення з системи координат пристрою у систему ENU.
   e. Обчислити штучні ознаки для моделі LSTM (4).
2. Прогнозування швидкості.
   a. Нормалізувати вхідні дані (нульове середнє і стандартне відхилення рівне одиниці).
   b. Отримати прогноз швидкості з LSTM на базі фіксованого вікна попередніх прискорень (2 секунди).

c. Розкласти швидкість на компоненти по осях *X* та *Y* знаючи напрям руху (5).

3. Обчислення координат.
   a. Обчислити швидкість на основі прискорення та відкоригувати її (6, 7).
   b. Обчислити зміщення по осі *X* та *Y*, інтегруючи відкориговану швидкість.

4. Візуалізація траєкторії.
   a. Перетворити отримані координати у геодезичну систему координат для відображення на карті.

Основною перевагою алгоритму є використання моделі LSTM на етапі прогнозування для компенсації дрейфу в інтегрованій швидкості. Етап попередньої обробки теж підвищує загальну стійкість, фільтруючи вхідні потоки та вирівнюючи часові мітки. Недоліком є слабко зв'язана архітектура нашої реалізації. Покладаючись на надані координати GNSS (та дешеве обладнання), ми втрачаємо можливості для оптимізації супутникової геолокації. Ще одним слабким місцем є залежність від магнітометра для визначення напрямку в горизонтальній площині (крок 2c).

Приклад фільтрування і перетворення прискорення наведено на рисунку 1:

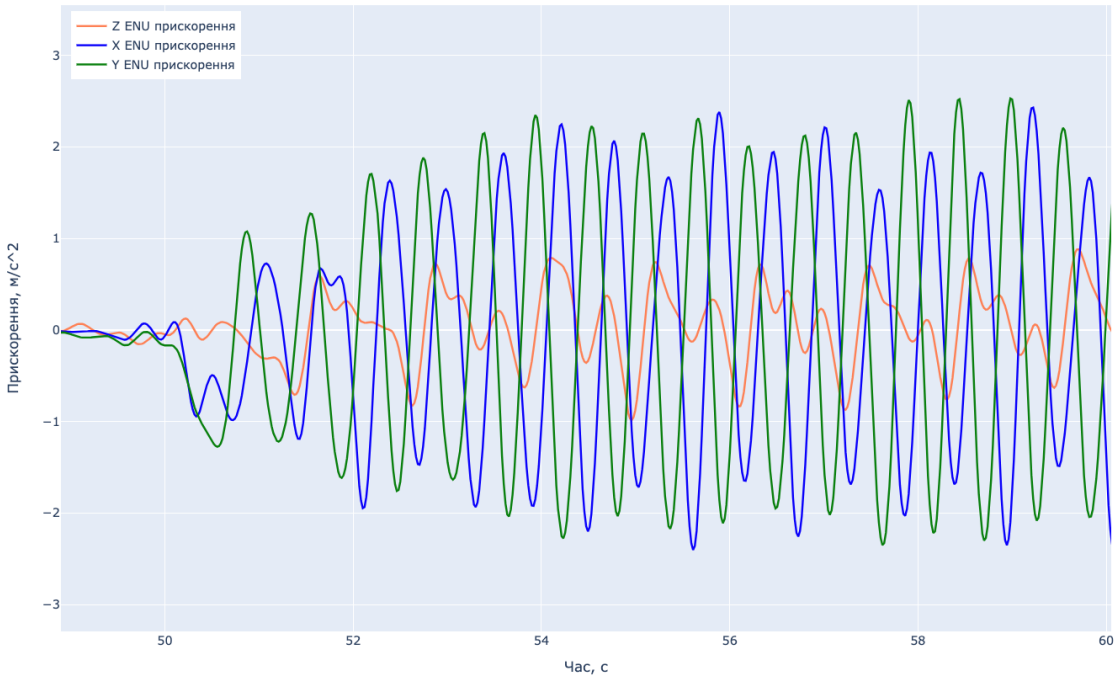

Рис. 1. Результат фільтрування сигналу з акселерометра (перші 10 секунд руху)

На кроці 1e для підвищення ефективності навчання LSTM ми обчислюємо додаткові ознаки: кумулятивне прискорення a, булевий прапорець m, що вказує на рух у вікні, та інтегровану швидкість v_z по осі Z:

$$a = \sqrt{a_x^2 + a_y^2 + a_z^2}$$
$$m = \begin{cases} 0, a < 0.25 \\ 1, a \geq 0.25 \end{cases} \quad (4)$$
$$v_z = v_z(t-1) + \int_{t-1}^{t} a_z(t)dt$$

На кроці 2 ми стандартизуємо вхідні дані з використанням тих самих параметрів, які були використані під час навчання моделі LSTM. Усі шість вхідних ознак повинні мати середнє значення рівне нулю, стандартне відхилення рівне одиниці. Ми використовуємо навчену модель LSTM для прогнозування швидкості v на кожній ітерації алгоритму. Вікно зі 106 вимірювань (2 секунди) обирається з потоку стандартизованих даних прискорення так, що вхідний тензор PyTorch має розмір 1×106×6.

Використовуючи кут напрямку з Android API, ми перетворюємо прогнозовану скалярну швидкість у векторну:

$$v_x = v * cos(yaw),$$
$$v_y = v * sin(yaw) \quad (5)$$

Як і в стандартному фільтрі Калмана контрольна матриця G реалізовує трапезоїдний метод інтегрування керуючих сигналів (прискорень) для оновлення стану. Він застосовується до контрольного вектора u, що містить поточне і попереднє прискорення a_k та a_{k-1}:

$$Gu = \begin{bmatrix} \frac{\Delta t^2}{4} & \frac{\Delta t^2}{4} \\ \frac{\Delta t}{2} & \frac{\Delta t}{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_k \\ a_{k-1} \end{bmatrix} \quad (6)$$

Але в нашому випадку, на етапі розрахунку пройденої відстані (крок 3) ми компенсуємо дрейф в інтегрованій швидкості (v_{int}) обчислюючи зважену суму з прогнозованою швидкістю:

$$v = \alpha v_{int} + (1 - \alpha) v_{pred} \quad (7)$$

Коефіцієнт $\alpha \in [0,1]$ дає змогу керувати злиттям двох потоків даних – від сенсора і від LSTM мережі. Формула (7) є простим комплементарним фільтром, який широко застосовується в галузі опрацювання сигналів для злиття потоків сенсорних даних.

Результати моделювання

Розглянемо детальніше результати навчання LSTM мережі. Експериментальні дані були зібрані за допомогою Android-смартфона середнього класу з використанням додатку Sensor Logger [2]. Набір даних №1 – це дані 4-х пішохідних маршрутів загальною довжиною до кількох кілометрів. Він містив 133 тис. рядків і був використаний для навчання та валідації LSTM моделі. Ідеальні погодні умови та відкрита видимість неба дозволили довіряти вимірюванням швидкості GNSS і обрати її як цільову змінну. Набір даних №2 містив п'ять маршрутів пішохода, який рухається строго прямокутною еталонною траєкторією, тримаючи телефон горизонтально і спрямовуючи його вперед. Прямокутна траєкторія *ABCD* мала розміри 77×15 метрів (рис. 4).

Всі набори даних з прискоренням містили такі стовпці, як:

- *timestamp*, абсолютний час заміру,
- *seconds elapsed*, час відносно початку сесії,
- *x*, прискорення вздовж осі *X*, м/с^2,
- *y*, прискорення вздовж осі *Y*, м/с^2,
- *z*, прискорення вздовж осі *Z*, м/с^2.

Для навчання найкращої моделі на наборі даних №1 ми дотримувалися стандартної процедури очищення даних, фільтрації шуму, інженерії нових ознак, поділу на навчальну, валідаційну та тестову вибірку, а також нормалізації даних за допомогою Standard Scaler. Було використано реалізацію мереж LSTM і RNN в PyTorch з 2 шарами, функцією втрат MSE і оптимізатором Adam. Розбиття на навчальну, валідаційну і тестову вибірки здійснено у пропорції 80/10/10. Розмір вхідного пакету, розмір прихованого шару та коефіцієнт відсіювання (dropout) були обрані як гіперпараметри. Для контролю швидкості навчання розмір пакету встановлювався рівним подвійному розміру прихованого шару. Швидкість навчання становила 0.0005, а кількість епох – 10.

Таблиця 1 підсумовує результати навчання. Очікувано, RNN не показала переваги в точності над LSTM, тільки у швидкості навчання. Що цікаво, менший розмір прихованого шару в мережі LSTM дав кращий результат. Це можна пояснити відносною простотою задачі регресії швидкості на базі періодичних закономірностей руху, прихованих у даних прискорення. Для вивчення цієї залежності потрібен був лише мінімальний обсяг "пам'яті".

Таблиця 1

Оцінки якості навчених моделей

Модель	Прихований шар	Кістьсть ознак	MAE	RMSE	R^2
LSTM-16-L-0.3	16	6	0.09843	0.12246	0.80489
LSTM-16-L-0.1	16	6	0.08692	0.11462	0.82908
LSTM-32-L-0.3	32	6	0.09820	0.12235	0.80523
LSTM-32-L-0.2	32	6	0.10743	0.13209	0.77298
LSTM-64-L-0.3	64	3	0.15988	0.18603	0.54976
LSTM-64-L-0.3	64	6	0.11468	0.14138	0.73994
RNN-16-L-0.1	16	6	0.15260	0.18640	0.54810
RNN-32-L-0.1	32	6	0.12570	0.15760	0.67670

Рис. 2. Порівняння прогнозованої та виміряної швидкості (MAE = 0.08692)

Для демонстрації роботи моделі, ми застосували її для прогнозування однієї з траєкторій з набору даних №2. Ми перетворили скалярну швидкість, отриману з LSTM, у вектор швидкості, використовуючи кут орієнтації *yaw* (5). Як і очікувалося, прогнозована швидкість не мала видимого дрейфу на більших відстанях (див. рис. 3). Інтегруючи прогнозовану швидкість, було обчислено координати *X*, *Y* у системі ENU та отримано чисто "інерційну" траєкторію (рис. 4).

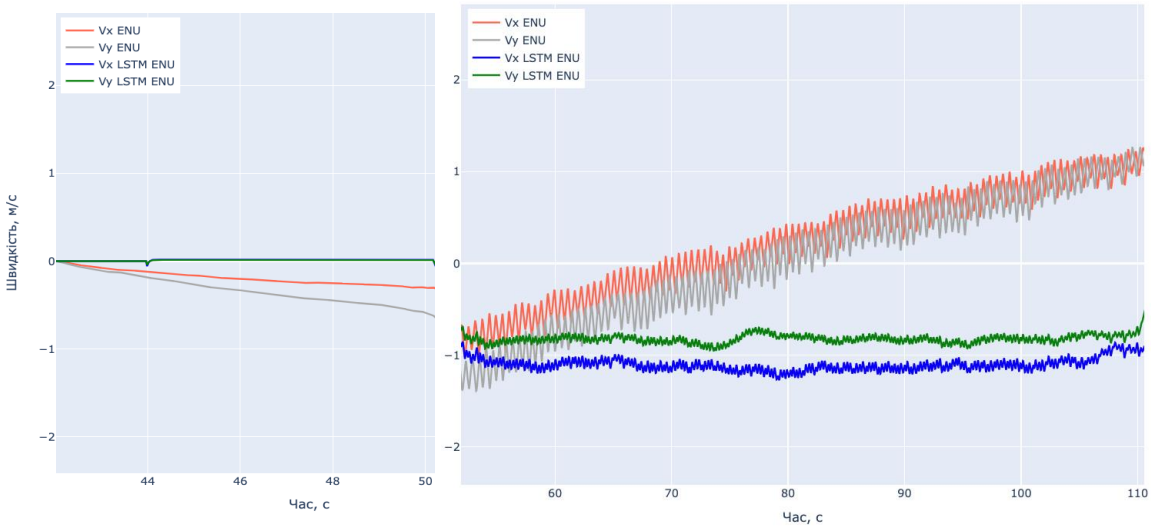

Рис. 3. Прогнозована та інтегрована швидкість в стані спокою (зліва) і в русі (справа)

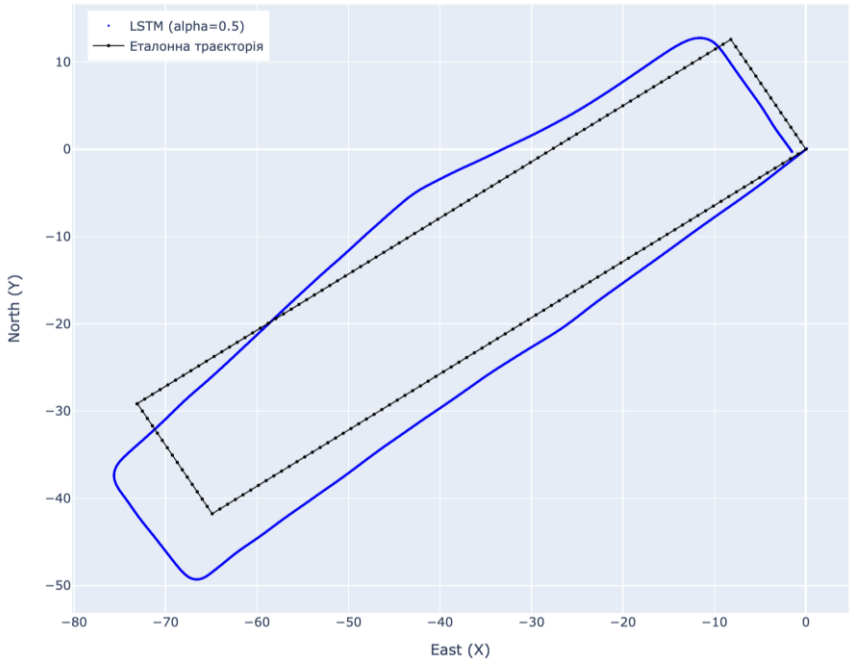

Рис. 4. Еталонний маршрут 77×15 м. та інерційна траєкторія на базі прогнозу LSTM

Висновки

Методи і засоби навігації в міському середовищі та в приміщеннях станом на сьогодні не є досконалими, особливо у випадку застосування звичайних смартфонів. Для підвищення точності, деякі з найкращих рішень використовують злиття потоків даних з багатьох сенсорів, а також методи машинного навчання.

У статті було показано, що IMU сенсори смартфона не забезпечують достатню точність для прямого обчислення швидкості і позиції на основі прискорення, потрібен метод для усунення дрейфу. Використання попередньо натренованої моделі LSTM для прогнозування швидкості дало змогу усунути дрейф та відтворити приблизну траєкторію руху в задачі навігації пішохода. Незважаючи на досягнення хорошої якості моделі LSTM (*MAE* = 0.087, R^2 = 0.83), для повного розкриття потенціалу цього підходу потрібні подальші уточнення орієнтації в горизонтальній площині. Щоб використати покази додаткових сенсорів, оцінки якості побудованої LSTM моделі слід використати для ініціалізації матриць коваріантності у фільтрі Калмана, і коригувати обчислену траєкторію показами GNNS сенсора.

Література

1. Asaad S., Outdoor Localization Solutions in IoT era: Research Challenges and Future Perspectives. [Електронний ресурс] / Asaad S., Maghdid H. // TechRxiv, – 2021. – Режим доступу: https://www.techrxiv.org/doi/full/10.36227/techrxiv.15138609 (дата звернення: 10.08.2024). – Назва з екрана.

2. Choi K., Sensor Logger [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/tszheichoi/awesome-sensor-logger (дата звернення: 02.08.2024). – Назва з екрана.

3. Niu Z., RTK with the Assistance of an IMU-Based Pedestrian Navigation Algorithm for Smartphones. / Niu Z., Nie P., Tao L., Sun J., Zhu B. // Sensors, – Vol. 19, – Issue 14, – 2019.

4. Jeong M., Comparison of Drift Reduction Methods for Pedestrian Dead Reckoning Based on a Shoe-Mounted IMU. / Jeong M., Cho S., Lim H. // Journal of Sensor Science and Technology, – Vol. 28, – Issue 6, – 2019.

5. MapIV, eagleye. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/MapIV/eagleye (дата звернення: 10.08.2024). – Назва з екрана.

6. Shen Z., A LiDAR-IMU-GNSS Fused Mapping Method for Large-Scale and High-Speed Scenarios. / Shen Z., Wang J., Pang C., Lan Z., Fang Z. // Measurement, – Vol. 225, – 2024.

7. Chen H., A GNSS/LiDAR/IMU Pose Estimation System Based on Collaborative Fusion of Factor Map and Filtering. / Chen H., Wu W., Zhang S., Wu C., Zhong R. // Remote Sensing, – Vol. 15, – Issue 3, – 2023.

8. Aggarwal G., Precise Indoor Positioning Without GPS Using a Device Mesh. / Aggarwal G., Kurinchi-Vendhan A., Wang A., Brar K. // Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), – 2017.

9. Zhao H.-Y., Smartphone-Based 3D Indoor Pedestrian Positioning Through Multi-Modal Data Fusion. / Zhao H.-Y., Cheng W., Yang N., Qiu S., Wang Z., Wang J. // Sensors, – Vol. 19, – Issue 20, – 2019.

10. Yan H., RoNIN: Robust Neural Inertial Navigation in the Wild—Benchmark, Evaluations, and New Methods. / Yan H., Herath S., Furukawa Y. // arXiv preprint arXiv:1905.12853, – 2019.

11. Yan H., RIDI: Robust IMU Double Integration. / Yan H., Shan Q., Furukawa Y. // In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds.) Computer Vision – ECCV 2018. LNCS 11217, Springer, – 2018.

12. Coskun H., Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. / Coskun H., Achilles F., DiPietro R., Navab N., Tombari F. // In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), – 2017.

13. Simon D., Kalman Filtering. / Simon D. // Embedded Systems Programming, – Vol. 14, – Issue 6, – 2001.

References

1. Asaad S., Outdoor Localization Solutions in IoT era: Research Challenges and Future Perspectives. [Electronic resource] / Asaad S., Maghdid H. // TechRxiv, – 2021. – Mode of access: https://www.techrxiv.org/doi/full/10.36227/techrxiv.15138609 (date of access: 10.08.2024). – Title from screen.

2. Choi K., Sensor Logger. [Electronic resource] // GitHub. – Mode of access: https://github.com/tszheichoi/awesome-sensor-logger (date of access: 02.08.2024). – Title from screen.

3. Niu Z., RTK with the Assistance of an IMU-Based Pedestrian Navigation Algorithm for Smartphones. / Niu Z., Nie P., Tao L., Sun J., Zhu B. // Sensors, – Vol. 19, – Issue 14, – 2019.

4. Jeong M., Comparison of Drift Reduction Methods for Pedestrian Dead Reckoning Based on a Shoe-Mounted IMU. / Jeong M., Cho S., Lim H. // Journal of Sensor Science and Technology, – Vol. 28, – Issue 6, – 2019.

5. MapIV, eagleye. [Electronic resource] // GitHub. – Mode of access: https://github.com/MapIV/eagleye (date of access: 10.08.2024). – Title from screen.

6. Shen Z., A LiDAR-IMU-GNSS Fused Mapping Method for Large-Scale and High-Speed Scenarios. / Shen Z., Wang J., Pang C., Lan Z., Fang Z. // Measurement, – Vol. 225, – 2024.

7. Chen H., A GNSS/LiDAR/IMU Pose Estimation System Based on Collaborative Fusion of Factor Map and Filtering. / Chen H., Wu W., Zhang S., Wu C., Zhong R. // Remote Sensing, – Vol. 15, – Issue 3, – 2023.

8. Aggarwal G., Precise Indoor Positioning Without GPS Using a Device Mesh. / Aggarwal G., Kurinchi-Vendhan A., Wang A., Brar K. // Proceedings of the 2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), – 2017.

9. Zhao H.-Y., Smartphone-Based 3D Indoor Pedestrian Positioning Through Multi-Modal Data Fusion. / Zhao H.-Y., Cheng W., Yang N., Qiu S., Wang Z., Wang J. // Sensors, – Vol. 19, – Issue 20, – 2019.

10. Yan H., RoNIN: Robust Neural Inertial Navigation in the Wild—Benchmark, Evaluations, and New Methods. / Yan H., Herath S., Furukawa Y. // arXiv preprint arXiv:1905.12853, – 2019.

11. Yan H., RIDI: Robust IMU Double Integration. / Yan H., Shan Q., Furukawa Y. // In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds.) Computer Vision – ECCV 2018. LNCS 11217, Springer, – 2018.

12. Coskun H., Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. / Coskun H., Achilles F., DiPietro R., Navab N., Tombari F. // In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), – 2017.

13. Simon D., Kalman Filtering. / Simon D. // Embedded Systems Programming, – Vol. 14, – Issue 6, – 2001.