https://www.sworldjournal.com/index.php/swj/article/view/swj32-01-006
DOI: 10.30888/2663-5712.2025-32-01-006

УДК 681.5(042.3)

TWO-STAGE DYNAMIC TRACKER FOR TRACEBILITY OF THE MOVING OBJECTS

ДВОЕТАПНИЙ ДИНАМІЧНИЙ ТРЕКЕР ВІДСТЕЖУВАННЯ РУХОМИХ ОБ`ЄКТІВ

Semko Oleksiy Viktorovich / Семко О.В.
к.т.н., с.н.с.
ORCID: 0000-0001-6473-1329
Інститут телекомунікацій і глобального інформаційного простору НАН України,
Київ, Чоколівський бульвар, 13, 03186
Vinnichuk Dinys Oleksandrovich / Віннічук Д.О.
ORCID: 0009-0001-9441-7696
Національний університет біоресурсів і природокористування України,
Київ, вул.. Героїв Оборони, 15, 03041

Анотація. *Відстеження об'єктів на відео є складним завданням комп'ютерного зору - CV (computer vision). Разом з тим, існує ряд проблем, які ускладнюють точне та надійне відстеження об'єктів у відео потоках, а саме: оклюзія (перекриття об'єктів, зміни зовнішнього вигляду об'єктів, швидке переміщення об'єкті, зміни масштабу та перспективи, складний або динамічний фон, вихід об'єктів за межі кадру, обмеження в обчислювальних ресурсах.*

В статті розглянуто основні принципи відстеження об`єктів між кадрами на основі сигналів, що генерує згорткова нейронна мережа - CNN (Convolutional Neural Network), проаналізовано основні недоліки існуючих систем, Для вирішення задачі оклюзії запропоновано двоетапний трекер з динамічним пороговим значенням, який здатен відстежувати об`єки на відеозорбаженні в режимі реального часу. За результатами досліджень показано, що запропонований трекер демонструє прийнятні результати роботи на складних динамічних сценах.

Результати досліджень є корисними для широкого кола фахівців, які використовують технології CV.

Ключові слова: *оклюзія, згорткова нейронна мережа, трекер, модель, розпізнавання образів, класифікація.*

Вступ

У сучасних умовах стрімкого розвитку цифрових технологій *CV* виступає однією з найдинамічніших та найперспективніших галузей прикладного штучного інтелекту. Його основною метою є автоматизація процесу отримання, інтерпретації та аналізу зображень і відеоданих, що дозволяє системам машинного навчання наближатися до рівня людського сприйняття візуальної інформації.

Актуальність *CV* зумовлена зростаючим попитом на системи автономного прийняття рішень у широкому спектрі застосувань. Зокрема, технології *CV* є

ключовими в розвитку безпілотного транспорту, робототехніки, розумного відеоспостереження, медичної діагностики, промислової автоматизації, агротехнологій та доповненої реальності. Завдяки здатності забезпечувати високу точність виявлення, класифікації та відстеження об'єктів у реальному часі, ці системи значно підвищують ефективність як виробничих процесів, так і сервісних послуг [1].

Методи *CV* поділяються на декілька ключових напрямів, які охоплюють як класичні алгоритмічні підходи, так і сучасні методи, засновані на глибокому навчанні. До появи глибоких нейронних мереж *CV* базувався переважно на математичних моделях, які реалізовували алгоритмічну обробку зображень (каскади Хаара, гістограми орієнтованих градієнтів). На сучасному етапі найбільшої ефективності в задачах *CV* досягають методи глибокого навчання, зокрема згорткові нейронні мережі - *CNN* (*Convolutional Neural Networks, CNN*) [2]. *CNN* є спеціалізованим типом нейронних мереж, розробленим для обробки даних зі структурованою сіткою (наприклад, зображень). Основними структурними елементами *CNN* є:

- згорткові шари, які виконують фільтрацію зображення, виділяючи локальні патерни;
- шари активації, які вводять нелінійність;
- пулингові шари, які зменшують розмірність представлення, зберігаючи найважливішу інформацію;
- повнов`язні шари, які відповідають за класифікацію на основі векторного представлення зображення.

Нейромережі виявлення об'єктів виконують дві задачі - локалізацію об'єктів та класифікацію. Основні підходи до побудови архітектур можна розділити на два напрямки - одноетапні моделі і двоетапні моделі.

Одноетапні моделі *YOLO (v1–v11, SSD*) у свою чергу здійснюють локалізацію та класифікацію в межах єдиного проходу мережі, що дозволяє вирішувати задачі *CV* у реальному масштабі часуі. Такі моделі дозволяють швидко обробити велику кількість кадрів зображення, однак мають нижчу

точність розпізнавання в порівнянні з двоетапними моделями. *YOLO* є ефективним методом детекції об'єктів завдяки унікальному підходу, який формулює задачу як єдину задачу регресії, що дозволяє досягти високої швидкості та точності в реальному часі [4]. При використанні моделі *YOLO* на вхід нейронної мережі подається зображення, а вихідними сигналами є координати рамки об'єкта, клас об'єкта і ймовірність його знаходження у рамці.

Двоетапні моделі *R-CNN, Fast R-CNN* і *Faster R-CNN* [3] використовують підхід, перший етап якого полягає у генерації регіонів-претендентів (Region Proposals), а другий — у класифікації кожного з них за допомогою *CNN*. Застосування двоетапних моделей забезпечують високу точність детекції об’єктів на зображеннях за умов використання обчислювальних ресурсів.

Ключовими елементами архітектури *YOLO* є магістраль, *Neck* та вихідний класифікатор. Магістраль отримує ознаки з зображення за допомогою загорткових шарів та пулінгу. *Neck* комбінує ознаки з різних рівнів, використовуючи збільшення розмірності (апсемплінг) та інформацію не тільки з попереднього шару, а і бічні зв`язки з магістралі. Таким чином, використання архітектури *YOLO* дозволяє ефективно вирішувати задачі детекції і класифікації об’єктів різних розмірів. Класифікатор передбачає визначення для кожного об’єкта в кожній клітинці зображення координат (*x, y, w, h*), класу об'єкта, ймовірність класу об'єкта (рис. 1).

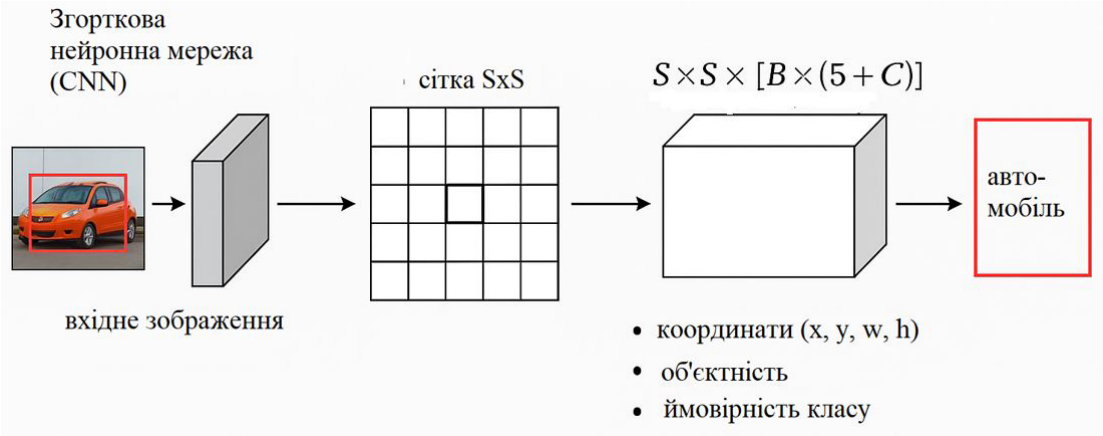

Рисунок 1 – Архітектура нейронної мережі

Моделі архітектури *YOLO* (рис.1) розпізнають положення об`єкту на конкретному кадрі. Для того, щоб визначити фізичні характеристики об’єкту, наприклад вектор руху, потрібно мати інформацію про стан об’єкту спостереження протягом деякого часу. Таким чином, виникає задача зіставлення розпізнаного об`єкту з сигналами нейронної мережі, що були отримані на попередніх ітераціях аналізу просторових даних (сцен).

Основний текст

Вирішення задачі трекінгу об’єктів полягає у визначенні їх положення на послідовності кадрів відеопотоку з метою побудови траєкторії переміщення. Існуючі підходи до вирішення задачі можна визначати відповідно їх кількості способу ініціалізації, використанням глибоких моделей, тощо [5].

Початково методи трекінгу ґрунтувались переважно на класичних алгоритмах обробки зображень, лінійній фільтрації та байєсівських моделях. В методах і моделях трекінгу набув широкого застовування фільтр Калмана [6], який за своєю сутністю є оптимальною рекурсивною процедурою оцінки для систем, які змінюються в часі та описуються лінійними стохастичними моделями з гаусівськими шумами. Таким чином, фільтр Калмана дозволяє прогнозувати положення об'єкта між кадрами відеопотоку, згладжувати шуми від недосконалостей детектора, оцінювати швидкість та прискорення об'єкта при відповідній моделі стану.

Одним з методів трекінгу об'єктів є метод *SORT* (*Simple Online and Realtime Tracking*). Сутність цього методу полягає у поєднанні детектора об'єктів (наприклад, *YOLO, Faster R-CNN*) з прогнозуванням руху та асоціацією треків для забезпечення ефективного відстеження об'єктів у відеопотоці в режимі реального часу (рис. 2).

Найпоширенішою парадигмою трекінгу є виконання детекції на кожному кадрі відеопотоку, Таким чином, трекер вирішує задачу асоціації між детекціями (*SORT, Deep Sort, ByteTracker*) [7]. Слід зазначити, що методи *SORT* і його покращена версія *Deep Sort*, який враховує векторні характеристики (*appearance features*) об’єкту, використовуючи *CNN,* забезпечують швидку детекцію,

простоту реалізації в обчислювальних системах і дозволяють використовувати різноманітні детектори об’єктів.

Рисунок 2 – Схема роботи алгоритму трекінгу об’єктів *SORT*

Таким чином, можна виділити основні етапи роботи методів *SORT і Deep Sort* є:

1) детекція об'єктів з використанням окремого детектора (*YOLO*);

2) прогноз станів (застосування фільтра Калмана або аналогічної моделі);

3) асоціація (пошук відповідностей між треками і новими детекціями з використанням угорського алгоритму або косинусної подібності дескрипторів);

4) оновлення (корекція станів і оновлення треків);

5) ініціалізація/видалення треків (додавання нових треків або видалення втрачених).

Іншим підходом є поєднання детекції та трекінгу одночасно в межах єдиної моделі. Такі моделі засновані на трансформерах, які об’єднують просторово-часовий контекст та мають вбудовану пам’ять треків, дозволяючи уникнути окремої стадії асоціації.

Сучасні підходи до трекінгу використовують глибоко інтегровані архітектури з одночасною детекцією, асоціацією та прогнозуванням. Вибір

конкретного підходу залежить від вимог до точності, реального часу, стабільності треків та доступних обчислювальних ресурсів.

Таким чином, трекери забезпечують високий рівень точності в задачах мультиоб'єктного трекінгу (*MOT*) та функціонують у реальному масштабі часу. Разом з тим, незважаючи на прогрес у поєднанні глибоких нейромереж, фільтрів Калмана та асоціативних стратегій, сучасні трекери мають низку суттєвих недоліків, які обмежують їх ефективність в умовах завад і невизначеностей. Більшість трекерів функціонують за парадигмою *tracking-by-detection* і повністю залежать від якості попередньо виконаного детектування.

Трекер втрачає треки або створює нові помилкові якщо детектор пропускає об'єкти або видає хибні спрацьовування, ідентифікатори об'єктів часто змінюються у випадках близького розташування однакових об'єктів, тимчасової втрати об’єктів, поганої роздільності. Більшість трекерів використовують спрощену модель руху (лінійну з постійною швидкістю) у фільтрі Калмана. Це не дозволяє ефективно відстежувати об'єкти з нерівномірною або непередбачуваною траєкторією та не враховує впливу сцени. У разі тимчасової втрати об’єкту (наприклад, через оклюзію), трекер часто передчасно видаляє трек або ініціалізує новий трек після повторної появи об'єкту, створюючи розрив у ідентичності.

Разом з тим, більшість трекерів функціонують локально у кадрі та не враховують глобальну структуру сцени, часовий контекст на великій відстані та взаємодію між об’єктами. Трекингові системи, які використовують *CNN*-дескриптори (*Deep SORT, BoT SORT*), мають підвищену обчислювальну складність, що ускладнює роботу на вбудованих пристроях та обмежує масштабованість при великій кількості об'єктів.

Перспективними напрямками розвитку трекінг-систем є:

- інтеграція ReID-механізмів і пам’яті;
- контекстуально обізнані трекери;
- використання графових нейромереж для врахування взаємодій об'єктів;
- адаптивне навчання у нових умовах.

Виходячи з вищезазначеного, пропонується новий трекер, який забезпечує високу швидкодію, поєднує ідеї деяких існуючих трекерів, дозволяє врахувати зникнення об`єкту та дозволяє відстежувати об`єкти що мають високу швидкість переміщення.

У якості детектору обрано одностадійну *CNN* архітектури *YOLO*, а саме *YOLOv8*. Дана мережа дозволяє ефективно детектувати різноманітні об`єкти, а також демонструє високу швидкодію. Середня швидкість обробки кадру такою мережею складає 30-40 мілісекунд.

Основою трекеру є широковідомий підхід, коли детекції, які передала нейронна мережа та треки порівнюються кожен кадр. Сутністю алгоритму є те, що для кожної детекції здійснюється зіставлення існуючого треку, а у випадку відсутності треку створюється новий трек.

Кожен трек також проходить через лінійний фільтр Калмана, що дозволяє згладжувати траєкторію переміщення об’єкту і передбачати (прогнозувати) його рух, враховуючи завади і невизначеності. Стан об`єкту характеризується вектором параметрів

$$m_i = \left(c_x, c_y, s, r, \dot{c}_x, \dot{c}_y, \dot{s} \right), \forall m_i \subset M, i = \overline{1,n} \ , \qquad (1)$$

де c_x, c_y - координати центру об’єкту на зображенні, $\dot{c}_x = \frac{dc_x}{dt}, \dot{c}_y = \frac{dc_y}{dt}$ - похідні координат центру об’єкту на зображенні, s - площа рамки, r - співвідношення сторін рамки, $\dot{s} = \frac{ds}{dt}$ - похідна зміни співвідношення сторін рамки, M - множина станів об’єкту в просторі параметрів. m_i - вектор поточного стану об’єкту в просторі параметрів, n - кількість стані об’єкту при трекінгу. а станів об’єкту в просторі.

У відповідності до стану об’єкту (1) фільтр Калмана спрацьовує в два етапи. На першому етапі використовується модель руху об’єкту для передбачення його наступного стану:

$$\begin{cases} x_k = A^* x_{k-1} + B^* u \\ P_k = A^* P_{k-1} + A^T * Q' \end{cases} \qquad (2)$$

де x_k - прогнозований стан, A – матриця переходу стану, P_k - прогнозована матриця похибки, Q - шум процессу, B – матриця керування, u - вектор керування. У співвідношенні (2), що визначає трекінг об'єктів $u = 0$, за умов відсутності активного керування.

Після надходження нових значень параметрів вимірювання у відповідності з співвідношенням (1) фільтр оновлює прогноз, враховуючи ступінь довіри до моделі та точність вимірювання [8]:

$$\begin{cases} K = P_k * H^T * \left(H * P_k * H^T + R \right)^{-1} \\ x = x_k + K * \left(z - H * x_k \right) \\ P = \left(I - K * H \right) * P_k \end{cases}, \tag{3}$$

де z — нове вимірювання (детекція об'єкта), H — матриця спостереження, R — шум вимірювання, K — калманівське підсилення (вага нового вимірювання), x — оновлений стан.

Після виконання калманівського підсиленя (3) здійснюється зіставлення треків і детекцій. Для цього визначається метрика подібності з метою її максимізації або функція ціни для її подальшої мінімізації. У якості метрики співпадіння використовується *IoU* (Intersection over Union)

$$IoU = \frac{S_I}{S_U}, \tag{4}$$

де S_I - площа перекриття рамок, S_U – площа об`єднання рамок.

Слід зазначити, що такий підхід є недосконалим для об`єктів, які мають високу швидкість в кадрі і, як наслідок, можуть мати низьке значення *IoU* відповідно співвідношенню (4). Також низьке значення *IoU* відповідає випадку нелінійного руху об’єкту. В таких випадках фільтр Калмана не може коректно спрогнозувати положення об`єкта в наступному кадрі. Для вирішення проблеми доцільним є використання двоетапного підходу до зіставлення треків та детекцій.

При двоетапному підході на першому кроці використовується стандартна функція ціни, яка заснована на *IoU*:

$$M1 = 1 - IoU \tag{5}$$

Функція ціни (5) фактично визначає метрику неспівпадання. Таким чином, для рамок і треків які не перетинаються функція ціни $M1$ буде нульовою, а для статичних об`єктів, тобто таких, які не змінили своє положення в кадрі, вона буде дорівнювати одиниці. Для кожної пари детекція-трек обчислюється функція ціни. Таким чином будується матриця вартості. За допомогою угорського алгоритму оптимізації проходить зіставлення між треками та детекціями з мінімальною сумарною вартістю. Далі пари детекція-трек, які мають значення ціни менше ніж порогове, відбираються як такі, що були зіставлені. Для кращої адаптивності трекеру пропонується розраховувати порогове значення залежно від часу знаходження об`єкту в кадрі, а саме надавати пільгу трекам, що давно знаходяться на зображенні

$$\begin{cases} f = w_f + \eta \\ \eta = \begin{bmatrix} g^* \lambda, g * \lambda \le \eta_{\max} \\ \eta_{\max}, g * \lambda > \eta_{\max} \end{bmatrix} \end{cases}, \quad (6)$$

де f – порогове значення функції вартості, w_f - початкова порогова вартість, g – час знаходження треку на зображенні в кадрах, λ - швидкість підвищення порогового значення, $\eta_{\max}$ - максимальне значення коригуючого коефіцієнту порогового значення.

Тим самим, відповідно співвідношенню (6) модель “довіряє” об`єктам, які довше були кадрі, визначаючи їх як більш “надійні”.

Треки, для яких не знайдено детекцій передаються на наступний крок, на якому повторюється процес формування матриці вартості та оптимізації за допомогою угорського алгоритму за іншою функцією ціни

$$\begin{cases} M2 = \alpha * (1 - IoU) + \beta * \sigma \\ \alpha + \beta = 1 \end{cases}, \quad (7)$$

де $M2$ – метрика вартості, a – коефіцієнт врахування IoU, β – коефіцієнт врахування дистанції, σ – відстань між центрами детекції та треку.

Відповідно співвідношенню (7) відстань σ нормується на подвійний розмір рамки. У випадку, коли класи детекції і треків різні або значення відстані більше

одиниці, зіставлення не відбувається.

Використання такого підходу дає змогу врахувати випадки, коли об`єкт рухається швидко і значення $IoU$ доволі невелике. В такому разі алгоритм покладається на нормовану відстань від об`єкту.

Треки, які не були зіставлені на обох кроках, в найпростішому випадку мають видалятися з пам`яті. Слід зазначити, що такий підхід втрачає треки, якщо нейронна мережа на змогла розпізнати об'єкт на одному кадрі, тобто при "пропуску кадру". Для уникнення цієї проблеми вводиться пам`ять об`єкту, тобто трек зберігається для порівняння на наступних кадрах, а не видаляється відразу. Кількість кадрів, впродовж яких зберігається трек в пам`яті, можна визначити, як "життєздатність" треку. Параметр "життєздатності" χ залежить від часу існування ("віку") об'єкту та може бути формально визначеним

$$\begin{cases} \chi = \chi_0 + \mu \\ \mu = \begin{bmatrix} g * \varpi, g * \varpi \le \mu_{\max} \\ \mu_{\max}, g * \varpi > \mu_{\max} \end{bmatrix}, \end{cases} \quad (8)$$

де χ – значення функції "життєздатності", χ_0 – початкова "життєздатність", g – час знаходження треку на зображенні в кадрах, ϖ - швидкість підвищення, $\mu_{\max}$ – максимальне значення підвищуючого коефіцієнту "життєздатності".

Треки, для яких не було знайдено детекцій протягом періоду "життєздатності" видаляються з пам`яті. Впровадження такого підходу дозволяє динамічно зберігати об`єкти, які знаходяться більше часу на зображенні. Треки, які з`явились одноразово мають низький параметр "життєздатності" і відповідно швидко видаляються з пам`яті при зникненні.

Впровадження таких модифікацій, як динамічна "життєздатність" треку, динамічне порогове значення, а також двоетапне зіставлення дозволяє ефективніше детектувати швидкі об`єкти і враховувати часткове перекриття об`єктів (оклюзії).

Підсумки та висновки

Для дослідження властивостей запропонованого алгоритму двоетапного динамічного трекера відстежування рухомих об'єктів було використане

обчислювальне середовище *Apple M1 Pro* з технологією *Metal Performance Shaders* (*MPS*), яка інтегрована з *PyTorch,* високопродуктивною бібліотекою для оптимізації виконання складних обчислень з використанням графічних процесорі в середовищі програмування Python.

При проведенні імітаційного експерименту було визначено, що час обробки одного кадру трекером займає 4 мілісекунди, що дозволяє використувати запропонований трекер в режимі реального часу. У якості гіперпарметрів трекеру відповідно співвідношенням (6) і (8) було обране порогове значення функції вартості $f = 0.5$, коефіцієнт врахування *IoU* $\alpha = 0.5$, β – коефіцієнт врахування дистанції $\beta = 0.5$, максимальне значення коригуючого коефіцієнту порогового значення $\eta_{\max} = 0.2$, швидкість підвищення порогового значення $\lambda = 0.02$, початкова "життєздатність" $\chi_0 = 1$, швидкість підвищення $\varpi = 1$, максимальне значення підвищуючого коефіцієнту "життєздатності". Для оцінки ефективності трекерів було використано метрики точності мультиоб`єктного відтсежування *MOTA* (*Multiple Object Tracking Accuracy*) [9]

$$L = 1 - \frac{\sum_{i=1}^{N} \mathbb{R}_i + \mathbb{C}_i + \theta_i}{\sum_{i=1}^{N} \gamma_i}, \tag{9}$$

де L – значення метрики точності мультиоб`єктного відтсежування; $\mathbb{R}$ – кількість пропущених об'єктів (False Negatives), тобто об'єкти, які не були знайдені; $\mathbb{C}$ – помилкові спрацьовування (False Positives), тобто треки, які не відповідають жодному справжньому об`єкту; θ – кількість випадків, коли один об'єкт отримує різні *ID* об'єкту протягом часу; γ – загальна кількість об'єктів у всіх кадрах; i – нмер кадру; N – кількість кадрів.

Таким чином, *MOTA* є основною метрикою якості багатооб'єктного трекінгу, яка штрафує помилки, допущені трекером при пропуску об'єктів $\mathbb{R}$, помилкових трекінгах $\mathbb{C}$, зміні *ID* об'єкту під час трекінгу (*ID switches*). На рис. 3 наведено результат присвоєння *ID* об'єктам в результаті роботи трекера.

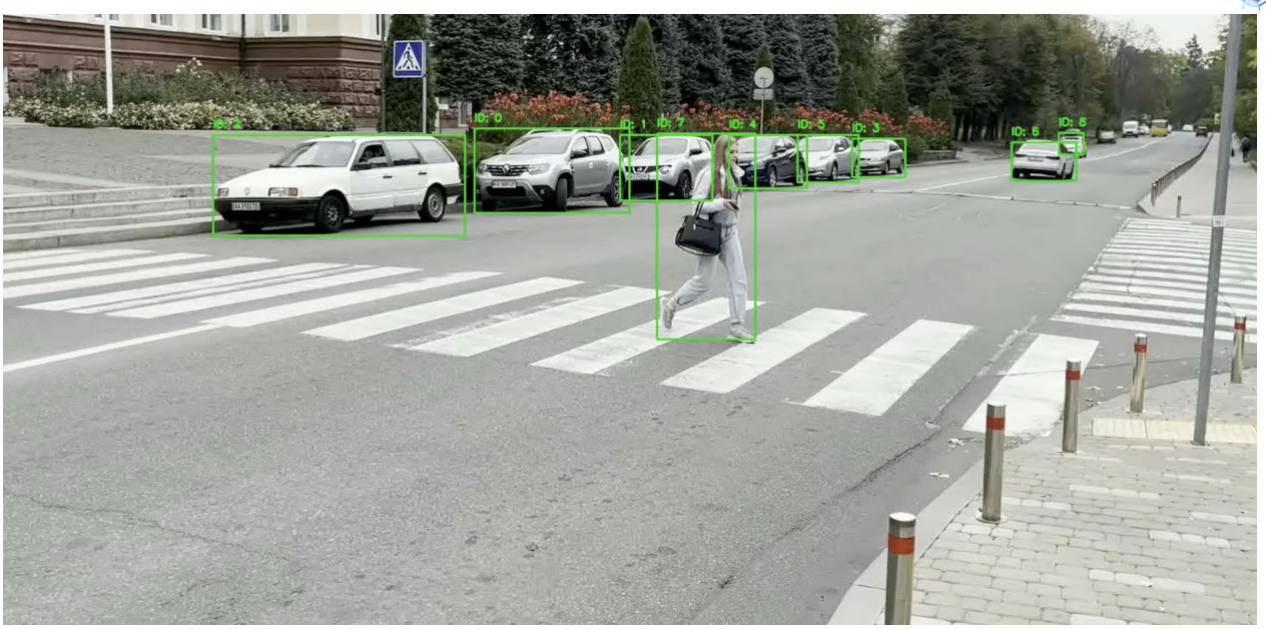

Рисунок 3 - Приклад роботи трекеру, а саме присвоєння ідентифікаторів об`єктам

Іншою метрикою, яка використовується для трекінгу є *IDF1* [10], яка враховує якість асоціації ідентифікаторів між предиктованими і справжніми об’єктами протягом усього відео.

$$IDF1 = \frac{2 * IDTP}{2 * IDTP + IDFP + IDFN} \quad , \tag{10}$$

де *IDTP* (*ID True Positives*) — кількість правильно відслідкованих об'єктів з існуючим *ID*; *IDFP* (*ID False Positives*) — помилкові присвоєння *ID*, яких не існує або визначається неправильна асоціація; *IDFN* (*ID False Negatives*) — об’єкти, яких трекер не відстежив.

На відміну від метрики *MOTA*, що фокусується на кількості об'єктів, метрика *IDF1* фокусується на якості *ID*-відстеження протягом часу. Якщо трекер правильно знаходить всі об'єкти, але часто плутає *ID*, то метрика *MOTA* буде високою, а показник *IDF1* буде низьким.

Для порівняння якості роботи створений детектор було порівняно з *ByteTrack* [11], який є надшвидким та ефективним трекером для множинного трекінгу об'єктів, що здобув популярність завдяки простоті реалізації та високій точності. Більшість трекерів, ігнорують об'єкти з низькою впевненістю. Детектор

ByteTrack асоціює навіть *low-confidence* рамки з існуючими треками, але тільки після того як рамки з високою впевненістю були враховані.

У якості тестових даних був обраний набір даних *The Vision Meets Drone Dataset* (*VisDrone*) [12]. Це великий реалістичний набір даних, які були зібраний з дронів, що використовується для вирішення задач *CV* (детекція об'єктів, трекінг, семантична сегментація, естимація глибини). Цей датасет також містить текстові файли з координатами та ідентифікаторами справжніх об’єктів, що дозволяє оцінити метрики трекерів.

На рис. 4 представлено кадр відео, на якому об’єкти зняті з різних висот, кутів та за різних умов освітлення.

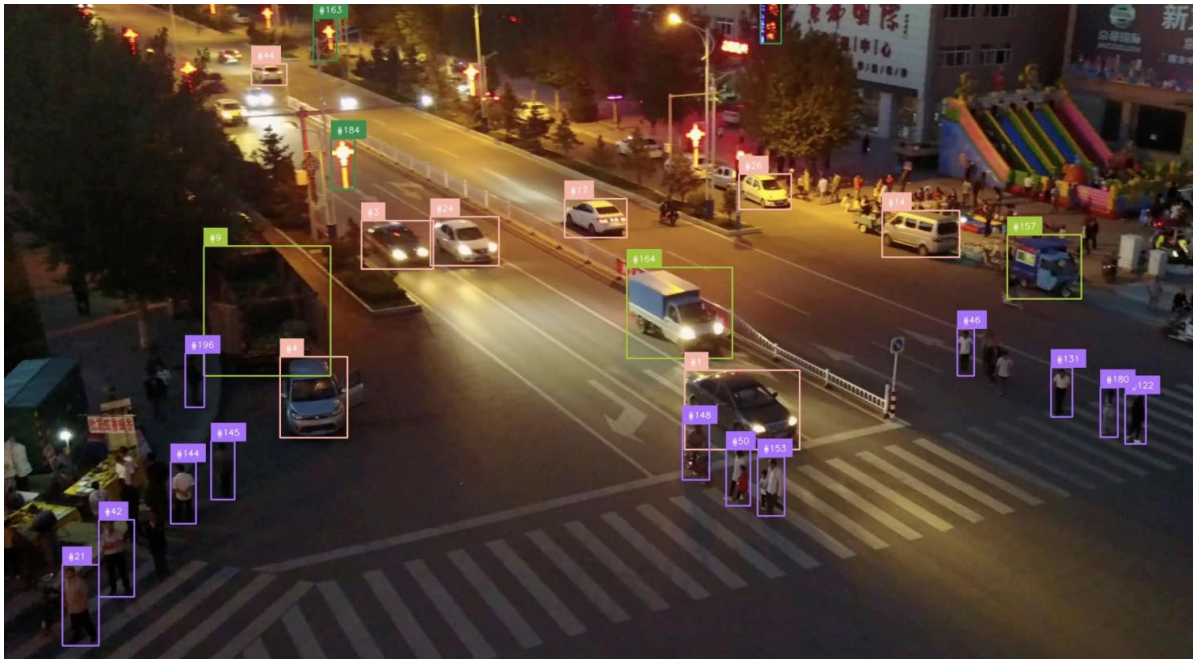

Рисунок 4 – Приклад роботи ByteTrack на кадрі з датасету VisDrone

На кадрі відео присутні часткові або повні оклюзії, зміни масштабу, нестабільна камера, велика кількість дрібних об’єктів. У випадку наявності швидкісних об’єктів спостереження, що переміщуються в тривірному просторі (літальні апарати) *VisDrone* одним з найскладніших *MOT*-датасетів (*Multiple Object Tracking* - відстеження кількох об'єктів).

В результаті моделювання ситуації, виходячи з відео, кадр якого наведено на (рис. 4), та вимірювань було отримані результати, які наведені в таблиці 1.

Таблиця 1 – Порівняння метрик MOTA та IDF1

Детектор	*MOTA*	*IDF1*
ByteTracker	0.41	0.5
Детектор, що був розроблений в ході роботи	0.54	0.52

Виходячи з результатів моделювання, можна зробити висновок про те, що значення метрики *MOTA* для запропонованого трекеру має суттєве покращення в порівнянні з трекером *ByteTrack*. Значення метрики *IDF1* демонструє несуттєву зміну, Тобто розроблений двоетапний трекер розпізнає об'єкти, що мають один і той самий ідентифікатор, на рівні трекера *ByteTrack*, але загалом відстежує більше об`єктів. Зауважимо, що для обох трекерів моделювання проводилось за допомогою детектору *YOLOv8x*.

За результатами проведеного дослідження запропоновано двоетапний трекер з динамічним пороговим значенням і "життєздатністю" треку, який працює в режимі реального часу та дозволяє ефективно відстежувати об`єкти, включаючи швидкісні і короткочасно зникаючі при оклюзіях. Зіставлення зображень в два етапи з використанням різних функцій ціни дозволяє врахувати як статичні, так і швидкісні об'єкти. Порівнюючи основні метрики запропонованого трекеру з існуючими рішеннями, можна зробити висновок про ефективність його роботи розробки. Швидкість обробки кадру у поєднанні з прийнятними значеннями метрик точності дозволяє використовувати запропонований трекер для вирішення практичних задач в тому числі і на інтегрованих малопотужних системах.

Література:

1. Majhi R. K., Waoo A. A. ADVANCES IN COMPUTER VISION: NEW HORIZONS AND ONGOING CHALLENGES. ShodhKosh: Journal of Visual and Performing Arts. 2024. Т. 5, № 5. URL: https://doi.org/10.29121/shodhkosh.v5.i5.2024.1893.

2. Understanding of Convolutional Neural Network (CNN): A Review / P. Purwono et al. International Journal of Robotics and Control Systems. 2023. Vol. 2,

no. 4. P. 739–748. URL: https://doi.org/10.31763/ijrcs.v2i4.888

3. Du L., Zhang R., Wang X. Overview of two-stage object detection algorithms. *Journal of physics: conference series*. 2020. Vol. 1544. P. 012033. URL: https://doi.org/10.1088/1742-6596/1544/1/012033

4. You only look once: unified, real-time object detection / J. Redmon et al. *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016. 2016. URL: https://doi.org/10.1109/cvpr.2016

5. A Review of Multi-Object Tracking in Recent Times / S. Li et al. IET Computer Vision. 2025. Vol. 19, no. 1. URL: https://doi.org/10.1049/cvi2.70010.

6. Taylor L. E., Mirdanies M., Saputra R. P. Optimized object tracking technique using Kalman filter. Journal of Mechatronics, Electrical Power, and Vehicular Technology. 2016. Vol. 7, no. 1. P. 57. URL: https://doi.org/10.14203/j.mev.2016.v7.57-66.

7. Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics / R. Pereira et al. *Applied Sciences*. 2022. Vol. 12, no. 3. P. 1319. URL: https://doi.org/10.3390/app12031319.

8. Велч Г., Бішоп Г. Вступ до фільтра Калмана [Електронний ресурс] / Г. Велч, Г. Бішоп.– Університет Північної Кароліни в Чапел-Гілл, Факультет комп'ютерних наук, 2006. – 16 с. – Режим доступу: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

9. Bernardin K., Stiefelhagen R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. EURASIP Journal on Image and Video Processing. 2008. Vol. 2008. P. 1–10. URL: https://doi.org/10.1155/2008/246309 .

10. Multiple object tracking: A literature review / W. Luo et al. Artificial Intelligence. 2020. P. 103448. URL: https://doi.org/10.1016/j.artint.2020.103448.

11. Zhang, Yifu & Sun, Peize & Jiang, Yi & Yu, Dongdong & Yuan, Zehuan & Luo, Ping & Liu, Wenyu & Wang, Xinggang. (2021). ByteTrack: Multi-Object Tracking by Associating Every Detection Box. URL: https://doi.org/10.48550/arXiv.2110.06864.

12. VisDrone-DET2020: The Vision Meets Drone Object Detection in Image Challenge Results / D. Du et al. SpringerLink. URL: https://doi.org/10.1007/978-3-030-66823-5_42.

Abstract. *Object tracking in video is a challenging task in computer vision (CV). At the same time, there are a number of problems that complicate accurate and reliable tracking of objects in video streams, namely: occlusion (overlapping objects, changes in the appearance of objects, fast object movement, changes in scale and perspective, complex or dynamic background, objects moving out of the frame, and limitations in computing resources.*

The article considers the basic principles of tracking objects between frames on the basis of signals generated by convolutional neural network (CNN), analyses the main shortcomings of existing systems, To solve the problem of occlusion, a two-stage tracker with a dynamic threshold value is proposed, which is able to track objects in the video image in real time. The research results show that the proposed tracker demonstrates acceptable performance on complex dynamic scenes.

Key words: *occlusion, convolutional neural network, tracker, model, pattern recognition, classification*