Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«ВІДСТЕЖЕННЯ РУХОМИХ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ З ЗАСТОСУВАННЯМ ФІЛЬТРУ КАЛМАНА»**

Виконав: студент 4-го року навчання,
Спеціальності
121 Інженерія програмного забезпечення
Марченко Владислав Олегович

Керівник Бучко О.А.
кандидат технічних наук, доцент

Рецензент _____

Кваліфікаційна робота захищена з оцінкою
_____

Секретар ЕК _____
«____» _____ 2025 р.

Київ – 2025

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

ЗАТВЕРДЖУЮ
Керівник кваліфікаційної роботи
_____ доцент Бучко О.А.
(підпис)
«____» _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту 4-го року навчання, освітній ступінь бакалавр, факультету інформатики

Марченко Владиславу Олеговичу

ТЕМА: Відстеження рухомих об’єктів у відеопотоці з застосуванням фільтру Калмана

ЗМІСТ ТЧ до кваліфікаційної роботи:

Зміст
Анотація
Вступ
1. Теоретичні аспекти роботи
2. Аналіз алгоритмів
3. Реалізація практичної частини
Висновки
Список літератури

Дата видачі «____» _____ 2025 р. Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Календарний план виконання роботи

Тема: «Відстеження рухомих об’єктів у відеопотоці з застосуванням фільтру Калмана»

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на кваліфікаційну роботу	08.10.2024	
2.	Пошук та огляд літератури	16.01.2025	
3.	Визначення структури практичної частини та створення прототипів	23.02.2025	
4.	Вивчення альтернативних алгоритмів, пошук та дослідження методів покращення алгоритму	07.03.2025	
5.	Оцінка алгоритму, тестування	20.03.2025	
6.	Фінальне оформлення практичної частини	05.04.2025	
7.	Створення презентації та написання доповіді	30.04.2025	
8.	Передзахист роботи	19.05.2025	
9.	Захист роботи	02.06.2025	

Студент *Марченко Владислав Олегович*

Керівник *Бучко Олена Андріївна*

«____» _____

Зміст

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

SORT – Simple Online and Realtime Tracking (Метод швидкого відстеження об’єктів у реальному часі)
DeepSORT – Deep Simple Online and Realtime Tracking (що використовує глибоке навчання для ідентифікації об’єктів)
OC-SORT – Observation-Centric SORT (Покращений метод SORT, орієнтований на спостереження над об’єктами)
IoU – Intersection over Union (Відношення площі перетину до площі об’єднання)
ReID – Re-Identification (Ідентифікація об’єкта при повторному спостереженні)
SOT – Single-Object Tracking (Однооб’єктне відстеження)
MOT – Multi-Object Tracking (Відстеження множини об’єктів)
SSD – Single Shot Detector (Детектор з одним проходом)
ORU – Observation-Centric Re-Update (компонента, що відповідає за оновлення треків після оклюзій)
OCM – Observation-Centric Momentum (Метод врахування руху, сфокусований на спостереженнях)
CNN – Convolutional Neural Network (Згорткова нейронна мережа)
R-CNN – Region-based CNN (Згорткова нейронна мережа, орієнтована на області зображення)
YOLO – You Only Look Once (Метод одноразового аналізу зображення для детекції об’єктів)
ROI – Regions of Interest (Передбачувані зони інтересу, де можуть знаходитись об’єкти)
EKF – Extended Kalman Filter (Розширений фільтр Калмана)
UKF – Unscented Kalman Filter (Фільтр Калмана, який використовує спеціальний набір точок для точнішої оцінки стану нелінійних систем)
NWD – Normalized Wasserstein Distance (Нормалізована відстань Васерштейна)

АНОТАЦІЯ

Робота присвячена дослідженню та розробці алгоритму відстеження об’єктів у відеопотоці, використовуючи фільтр Калмана та орієнтацію на спостереження над об’єктами (OC-SORT). Метою роботи є виявлення методів поліпшення стабільності менеджменту ідентифікаторів об’єктів у складних умовах, таких як оклюзія та нелінійний рух.

Основна ідея алгоритму полягає у впровадженні буфера спостережень для відновлення треків після втрати об’єктів та використанні механізму повторної корекції за допомогою останніх достовірних даних. Такий підхід дозволяє зменшити похибку та підвищити точність відстеження об’єктів, особливо у випадках, коли стандартний фільтр Калмана втрачає об’єкт через тривалу його відсутність.

Перший розділ охоплює сучасні підходи з виявлення та трекінгу об’єктів, включаючи методи диференціювання кадрів (Frame differencing), оптичного потоку (Optical Flow), SSD, YOLO, прискорені R-CNN, а також алгоритми трекінгу, зокрема фільтр Калмана, частинковий фільтр (Particle Filter), DeepSORT, ByteTrack. У розділі також розглянуто основні обмеження фільтру Калмана та способи їх вирішення.

Другий розділ присвячено детальному аналізу проблем трекінгу у відеопотоці. Охоплено методи асоціації треків, зокрема IoU та Угорський алгоритм, а також використання метрик відповідності (Відстань Махаланобіса, NWD). Увагу приділено впливу налаштування трекера на стабільність ідентифікаторів.

У третьому розділі описано структуру та реалізацію алгоритму OC-SORT. Розглянуто архітектуру проєкту, принципи роботи компонентів. Проведено тестування алгоритму на основі методу диференціювання кадрів та YOLO, а результати порівняно з класичними реалізаціями трекерів на основі фільтру Калмана та детекцій.

Результати демонструють ефективність алгоритму OC-SORT у контексті менджменту ідентифікаторів об’єктів та стійкості до оклюзій, а також значне зниження кількості помилкових ідентифікацій у порівнянні з базовими підходами. Висновки містять рекомендації щодо подальшого розвитку алгоритму та можливостей його застосування у системах реального часу.

ВСТУП

В умовах розвитку технологій відеоаналітики та постійного зростання обсягів візуальних даних, актуальною є задача ефективного відстеження об'єктів у відеопотоці. ЇЇ метою є відслідковати рух і траєкторію кожного з них протягом подачі відео, що є основою для багатьох сучасних систем.

Основною проблемою традиційних алгоритмів є низька стабільність при оклюзіях та нелінійному русі об’єктів. Більшість з них використовують лінійну модель руху, що є прийнятним для коротких інтервалів часу, але стає недоліком у випадках тривалих оклюзій або різких змін траєкторій. Це призводить до похибки та втрати треків, що негативно впливає на цілісність траєкторій.

Актуальність теми дослідження зумовлена необхідністю підвищення точності відстеження об’єктів за складних умов, описаних вище. Сучасні алгоритми трекінгу, зокрема ті, що базуються на фільтрі Калмана, демонструють високу швидкість обробки даних, у той же час можуть бути вразливим до помилок у випадках відсутності детекцій. Це призводить до розриву треків або хибної асоціації об’єктів, що знижує загальну ефективність алгоритму.

Метою даної роботи є реалізація алгоритму трекінгу об’єктів на основі методу OC-SORT, який спрямований на усуненя накопичення помилок під час відсутності детекцій шляхом використання буфера спостережень та корекції треків при повторному виявленні об’єкта.

Для досягнення мети було поставлено такі завдання:

- Проаналізувати існуючі методи виявлення та відстеження об’єктів, визначити їх переваги та недоліки.
- Дослідити проблему асоціації об’єктів після оклюзії та визначити точки впливу на стабільність відстеження.
- Реалізувати алгоритм OC-SORT
- Виконати його тестування із застосуванням різних детекторів, а також провести порівняльний аналіз з іншими алгоритмами.

Наукова новизна роботи полягає у впровадженні механізму спостереження у процес трекінгу об’єктів, що дозволяє зменшити кількість помилок під час оклюзій та нелінійного руху.

Практичне значення полягає у можливості подальшого дослідження алгоритму на основі отриманих результатів, або його застосування у системах реального часу.

У роботі було оглянуто літературу, яка стосується сучасних методів детекції та відстеження рухомих об’єктів. Увагу приділено дослідженню підходів до корекції треків після оклюзій.

Методологічна основа роботи включає аналіз сучасних алгоритмів трекінгу, реалізацію алгоритму OC-SORT, а також тестування його ефективності на прикладі різних детекторів і порівняння з іншими алгоритмами.

Результатом є підвищення стабільності трекінгу за рахунок збереження спостережень у буфері та всоціація об’єктів після повторної детекції, що підвищує точність трекінгу в реальних умовах.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ВІДСТЕЖЕННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

1.1 Основи виявлення та відстеження об'єктів

Виявлення об'єктів полягає у визначенні їхніх координат на кожному кадрі відео. Результатом є набір обмежувальних рамок, які ідентифікують кожен об'єкт. Є два основних підходи до реалізації – традиційні детектори, які аналізують кадри математично, та нейромережеві детектори. До перших належать методи диференціювання кадрів, відокремлення фону, оптичеого потоку та інші, які аналізують піксельні зміни між послідовними кадрами. Нейромережві детектори, такі як YOLO, SSD та прискорені R-CNN, здатні не лише виявляти об'єкти, а й класифікувати їх.

Другим важливим етапом є відстеження (або трекінг) об'єктів, що забезпечує траєкторії кожного об'єкта. Воно може бути двох типів: однооб'єктне (SOT) або мультиоб'єктне (MOT). SOT зазвичай потребує на вхід рамку навколо об'єкта, який треба надалі відслідковувати, і тільки його єдиного. MOT, навпаки ж, не потребує жодних вхідних умов, достатньо лише сам потік кадрів з відео, при чому відслідковуватись мають всі можливі об'єкти одночасно, що є більш актуальним завданням у більшості сферах [1]. У даній роботі надалі буде розглядатись саме MOT.

Трекінг враховує попередні позиції об'єкта для прогнозування його подальшого руху. Основною метою тут є зберегти унікальний ідентифікатор для кожного об'єкта та оновлювати його координати з плином часу. При цьому, реальні умови дають певні перешкоди в цьому. Однією з основних проблем є оклюзія, коли об'єкт тимчасово перекривається іншими об'єктами або виходить за межі кадру. У таких випадках алгоритм повинен зберігати ідентифікатор об'єкта, навіть якщо його не видно. Зміна масштабу та різкі зміни напрямку руху також ускладднюють процес, оскільки можуть призводити до помилкових асоціацій між об'єктами та їхніми траєкторіями. Важливим є також стійкість алгоритму до хибних детекцій, до прикладу шум може сприйматися як нові об'єкти. Для вирішення у трекерах використовують методи фільтрації даних та корекції траєкторій [2].

1.2 Класифікація методів виявлення об'єктів

1.2.1 Диференціювання кадрів

Диференціювання кадрів (Frame differencing) є одним із простих та водночас ефективних методів виявлення рухомих об’єктів. Суть полягає у порівнянні кадру з попереднім для виявлення змін у піксельних значеннях. Зміни, що перевищують певний заданий поріг, вважаються потенційними рухомими об’єктами.

Спочатку здійснюється порівняння поточного та попереднього кадру шляхом обчислення абсолютної різниці між пікселями обох з них за формулою:

$$D(x, y) = |F_t(x, y) - F_{t-1}(x, y)|$$

де $D(x, y)$ – різниця яскравості пікселя у координатах (x, y),

$F_t(x, y)$ – яскравість пікселя на поточному кадрі,

$F_{t-1}(x, y)$ – яскравість пікселя на попередньому кадрі.

Для виділення областей, що містять рухомі об’єкти, йде порівняння з порогом. Це дозволяє відокремити рухомі об’єкти від фону. Визначення маски руху здійснюється за такою формулою:

$$M(x, y) = \begin{cases} 1, \text{якщо } D(x, y) > T \\ 0 \text{ в іншому випадку} \end{cases}$$

де T – порогове значення, яке підбирається емпірично або адаптивно залежно від рівня шуму та освітлення у кадрі.

Після цього йде фільтрація шуму за допомогою операцій, таких як ерозія та дилатація. Це дозволяє зменшити кількість хибних детекцій та об’єднати декілька областей, що відповідають одному об’єкту, але це працює не ідеально.

У кінці виділяються контури на основі отриманої маски руху, що використовуються для побудови обмежувальних рамок, які ідентифікують кожен об’єкт [3].

Диференціювання кадрів має такі переваги, як простота реалізації та висока швидкість обробки. Проте серед недоліків метод є чутливим до змін освітлення та неефективним у випадках швидкого руху об’єктів, складних змін фону і масштабних об’єктів.

1.2.2 Оптичний потік

Оптичний потік (Optical Flow) оцінює рух об'єктів на основі змін яскравості між послідовними кадрами відео. Рух пікселів можна розглядати як певний вектор, який вказує напрямок та швидкість. Метод має широке застосування у задачах виявлення об’єктів, трекінгу та аналізу руху.

Оптичний потік визначається як векторне поле, що описує зміну положення пікселів у просторі між двома сусідніми кадрами. Формально це можна виразити так [4]:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

де $I(x, y, t)$ – яскравість пікселя у кадрі t на координатах (x, y),

$\Delta x, \Delta y$ – зміщення пікселя по горизонталі та вертикалі,

Δt – часовий інтервал між двома кадрами.

Треба визначити вектори руху. Для цього використовуються часткові похідні за часом та координатами:

$$I_x u + I_y v + I_t = 0$$

де I_x, I_y – похідні за координатами x та y,

I_t – часткова похідна за часом,

u, v – горизонтальна та вертикальна швидеість.

У результаті маємо одне рівняння та дві невідомі, через що задача Optical Flow є недовизначеною. Для вирішення цієї проблеми існує декілька підходів, зокрема:

- Метод Лукаса-Канаде – аналізує групи сусідніх пікселів, вважаючи, що вони мають однаковий рух. Це зменшує вплив шуму.
- Метод Фарнебака – використовує поліноміальне наближення для оцінки векторів руху. Краще працює зі складними текстурами та дрібними деталями.

- Метод Хорна-Шунка – додає додаткове обмеження на плавність векторного поля, щоб уникнути різких стрибків у значеннях та зменшити вплив шуму.

1.2.3 Відокремлення фону

Відокремлення фону (Background Subtraction) порівнює поточний кадр з попередньо визначеною фоновою моделлю. Виділяє ті пікселі, які суттєво відрізняються від фону, що дозволяє ідентифікувати рухомі об'єкти.

Фонову модель формується за допомогою обчислення середнього значення інтенсивності пікселів за певний проміжок часу. Потім кожен новий кадр порівнюється з цією самою моделлю, і пікселі, що значно відрізняються від середнього значення, вважаються об'єктами, які з'явилися у сцені, і потім будуються рамки навколо них. З часом фонова модель оновлюється, щоб адаптуватися під зміни у відео.

Однієї реалізації алгоритмів формування, оновлення фонової моделі, порівняння кадрів із нею не існує. Реалізації є різні, серед прикладів:

- Рухоме середнє (Running Average) – виконуються обчислення середнього значення для кожного пікселя протягом певного часу.
- Метод K найближчих сусідів (K-Nearest Neighbors) – використовує класифікацію кожного пікселя на основі аналізу його найближчих сусідів у фоновій моделі.
- Модель гауссової суміші (Gaussian Mixture Model) – використовує набір гауссових розподілів для моделювання фону. Кожен піксель представляється сумішшю гауссових функцій, а оновлення фону здійснюється шляхом корекції параметрів моделей. Ефективний при зміні освітлення [5].

1.2.4 SSD

Детектори SSD представляють собою нейронну мережу для виявлення об’єктів, що використовує одноетапний підхід для прогнозування рамок та класів об’єктів. На відміну від двоетапних детекторів, SSD виконує обидві задачі одночасно, що підвищує швидкість.

Мають архітектуру, де кожен рівень відповідає за виявлення об’єктів різного масштабу. Шляхом використання різних розмірів фільтрів можливо одночасно виявляти і дрібні, і великі об’єкти [6].

SSD складається з наступних основних компонентів:

- Базова мережа (Base Network) – попередньо навчена нейронна мережа, яка виділяє основні просторові ознаки зображення.
- Додаткові шари (Extra Layers) – додаються після базової мережі для обробки отриманих ознак на різних рівнях масштабів.
- Прогнозувальні шари (Prediction Layers) – прогнозують обмежувальні рамки та класи об’єктів для кожного з рівнів.
- Функція втрат (Loss Function) – локалізаційна (для координат обмежувальних рамок) та класифікаційна (для визначення класу об’єкта).

SSD характеризуються високою швидкістю обробки відносно аналогів завдяки одноетапному підходу. Мають певні недоліки, зокрема знижену точність при виявленні дрібних об’єктів порівняно з двоетапними детекторами. Крім того, можуть демонструвати труднощі у виявленні об’єктів, які сильно прикриті чимось на кадрі.

1.2.5 YOLO

YOLO – сучасний алгоритм виявлення об'єктів, який використовує єдину нейронну мережу для одночасного виявлення та класифікації об'єктів. На відміну від традиційних підходів, які спочатку генерують регіони інтересу (ROI) та потім класифікують їх, YOLO робить це все одразу.

Розбиває зображення на сітку $S \times S$. Кожна комірка цієї сітки відповідає за виявлення об'єктів, центри яких потрапляють усередину неї. У кожній комірці мережа прогнозує [7]:

- Параметри обмежувальних рамок (центр x y, ширину w, висоту h).
- Найбільш імовірний розмір рамок (серед декількох наявних).
- Імовірність наявності об'єкта у комірці.
- Клас об'єкта (обмежений список класів, на якому навчена модель).

Архітектура:

- Вхідний шар – приймає зображення розміром $N \times N$ пікселів.
- Згорткові шари – визначають просторові ознаки об’єктів.

- Повнозв’язані шари – обчислюють вихідні параметри для кожної комірки сітки.
- Вихідний шар – повертає набір рамок, клас об'єкта та впевненість у детекції.

Результатом роботи повертається тензор розміром $S \times S \times (B \times 5 + C)$, де

S – розмір сітки, на яку ділиться зображення,

B – кількість обмежувальних рамок, які прогнозуються на кожну з комірок,

C – кількість класів у моделі,

5 – кількість параметрів для кожної рамки (x, y, w, h, confidence)

Функція втрат моделі включає три основні компоненти:

- Локалізаційна втрата – штраф за помилки у прогнозуванні параментрів рамок.
- Втрата довіри – штраф за помилки у прогнозуванні впевненості.
- Класифікаційна втрата – штраф за неправильну класифікацію об'єкта.

На даний момент YOLO є одним з найкращих інструментів для виявлення об’єктів на зображеннях, бо одночасно і детектить, і класифікую об’єкти, при чому має досить високу точність, а також відносно пристойну швидкість, що дозволяє використовувати модель в реальному часі. Серед недоліків: обмеження по типам об’єктів через те, що тренування здійснюється відповідно на обмеженій кількості класів. За потреби тренування нової моделі на інших даних є потреба в потужній відеокарті, що за відсутності також є недоліком.

1.2.6 Прискорені R-CNN

Прискорені R-CNN (Faster R-CNN) є удосконаленням попередніх версій R-CNN та швидких R-CNN (Fast R-CNN), зі зменшеним часом обробки та оптимізованою структурою нейромережі.

Складається з трьох основних компонентів [8]:

- Блок вилучення ознак – CNN, яка виділяє ознаки з вхідного зображення. Вихід цього блоку – це матриця отриманих ознак.
- Мережа пропозицій регіонів – нововведення у порівнянні з попередніми моделями. Є окремою невеликою нейромережею, яка сканує матрицю

ознак та генерує регіони інтересу (ROI) – це області, що ймовірно містять об’єкти.

- Агрегація ознак з областей інтересу та класифікація – на основі отриманих регіонів, витягуються матриці ознак фіксованих розмірів. Ці ознаки подаються до класифікатора, який визначає клас об’єкта та уточнює координати обмежувальної рамки.

Однією з переваг прискорених R-CNN є зменшення часу обробки у порівнянні з попередніми версіями завдяки мережі пропозицій регіонів. Однак, метод все ще досить вимогливий до обчислювальних ресурсів через використання двоетапної архітектури, особливо для зображень більшого розміру. У порівнянні з методами, такими як SSD або YOLO, має низьку швидкість.

1.3 Огляд сучасних алгоритмів трекінгу

1.3.1 Фільтр Калмана

Фільтр Калмана використовується для оцінювання стану динамічної системи в умовах вимірювань з похибками. Поєднує попередні оцінки з новими вимірюваннями, щоб дати точнішу оцінку стану об’єкта. У контексті трекінгу, прогнозує де буде об’єкт на основі попередніх кадрів.

Складається з двох основних етапів – прогнозування (predict) та оновлення (update). На етапі прогнозування використовує попередні оцінки для передбачення поточного стану. На етапі оновленя — коригує ці оцінки на основі нових вимірювань, враховуючи похибку.

Математична модель може бути описана за допомогою наступних рівнянь [9]:

- Прогнозування:

$$x_{k|k-1} = F \cdot x_{k-1} + B \cdot u_{k-1}$$
$$P_{k|k-1} = F \cdot P_{k-1} \cdot F^T + Q$$

де:

$x_{k|k-1}$ – прогнозований стан системи,

F – матриця переходу станів,

B – матриця керування,

u_{k-1} – вектор керування,

$P_{k|k-1}$ – прогнозована коваріаційна матриця,

Q – матриця шуму процесу.

- Оновлення:

$$K_k = P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1}$$
$$x_k = x_{k|k-1} + K_k \cdot (z_k - H \cdot x_{k|k-1})$$
$$P_k = (I - K_k \cdot H) \cdot P_{k|k-1}$$

де:

K_k – коефіцієнт Калмана (Kalman gain),

z_k – нове вимірювання,

H – матриця спостереження,

R – коваріаційна матриця вимірювань,

I – одинична матриця.

Основна перевага фільтру Калмана полягає у його здатності прогнозувати стан об’єкта за умови відсутності детекцій, що в свою чергу допомагає зберігати один і той самий ідентифікатор об’єкта протягом максимального часу. Це досягається шляхом використання матриці переходу станів F, яка враховує попередню швидкість і напрямок руху об’єкта. Але базується він на припущенні, що об’єкти рухаються виключно лінійно, що часто може бути обмеженням у випадках інших нелінійних траєкторій.

Недоліками є чутливість до неправильного задання матриць шуму процесу Q та шуму вимірюваннь R. У випадках сильних оклюзій або різких змін руху, прогнозована траєкторія більше відхиляється від реальної. Для вирішення цієї проблеми існують розширені варіанти фільтру Калмана, зокрема EKF і UKF [14],[15].

1.3.2 Частинковий фільтр

Частинковий фільтр (Particle Filter) оцінює стан системи на основі набору випадкових зразків (частинок), розподіленими заданим чином. Дозволяє відстежувати об'єкти навіть за умов нелінійної динаміки, що робить його ефективним для складних сцен.

На відміну від фільтру Калмана, зберігає множину можливих станів у вигляді частинок. Кожна така частинка представляє можливий стан об'єкта та має свою вагу, що відповідає її ймовірності.

Основні етапи роботи [10]:

- Ініціалізація – створюється набір частинок, розподілених випадковим чином навколо початкової позиції об'єкта. Ваги всіх початкових частинок рівні, тобто ймовірність перебування об'єкта у цих позиціях однакова.
- Прогнозування – у кожен момент часу кожна частинка переміщується відповідно до моделі руху. Це може бути як просте переміщення на основі попередньої швидкості, так і більш складна модель. До кожної частинки додається випадковий шум, щоб врахувати можливі відхилення у траєкторії.
- Оновлення – на основі нових спостережень оцінюється, наскільки кожна частинка відповідає дійсному стану об'єкта. Ваги оновлюються і частинки, які краще відповідають спостереженням, отримують більшу вагу, а ті, що сильно відрізняються – меншу.
- Нормалізація ваг – всі ваги частинок нормалізуються таким чином, щоб їхня сума дорівнювала одиниці.
- Ресемплінг – частинки з більшою вагою залишаються, а з меншою відкидаються, щоб сфокусувати обчислення на областях кадрів, де ймовірність нового знаходження об'єкта є вищою.
- Оцінка стану – визначається поточний стан об'єкта шляхом обчислення середнього значення усіх частинок – наближене значення нових місцезнаходження та швидкості об’єкта.

Можна сказати, що частинковий фільтр є досить корисним у задачах трекінгу, коли об'єкти можуть рухатися за складними або нелінійними траєкторіями. Наприклад, у випадках, коли об'єкт різко змінює напрямок або зникає за оклюзією, цей метод може адаптуватися до змін руху завдяки ресемплінгу.

1.3.3 DeepSORT

DeepSORT є покращенням алгоритму SORT, що використовує нейронні мережі для ідентифікації об'єктів і покращення асоціації треків. Вводиться компонента ReID, яка дозволяє ідентифікувати об'єкти не лише на основі просторових координат, але й за візуальними ознаками. Це дозволяє значно знизити кількість помилкових асоціацій у випадках перехрещеня траєкторій або оклюзій.

Основні компоненти [11]:

- Детекція об'єктів – використовується будь-який сучасний детектор об'єктів, наприклад, YOLO або прискорена R-CNN. Детектор повертає набір обмежувальних рамок для кожного об'єкта в кадрі.
- Фільтр Калмана – аналогічно з SORT, використовується для прогнозування траєкторії об'єкта.
- ReID – для кожного об'єкта формується вектор ознак за допомогою нейронної мережі. Він описує візуальні характеристики об'єкта. В результаті, навіть якщо об'єкт тимчасово зникає з кадру або зазнає оклюзії, то після його повторної детекції будуть причини вважати його собою.
- Асоціація треків – спочатку об'єкти асоціюються на основі координат та фільтру Калмана за допомогою IoU метрики. Застосовується ReID вектор для покращення асоціацій.
- Оновлення треків – фільтр Калмана оновлює стан об'єктів та їхні вектори ознак. Вектори ознак зберігаються в буфері для подальшого порівняння у випадку втрати об'єкта.

У порівнянні з класичним SORT метод є набагато ефективнішим, бо дозволяє відстежувати об’єкти у складних умовах з оклюзіями та перехрещенням траєкторій. Недоліком є зростання обчислювальної складності, тому при обмежених ресурсах у реальному часі іноді неможливий до застосування.

1.3.4 ByteTrack

ByteTrack є покращенням SORT та DeepSORT. Основна ідея полягає у використанні невпевнених детекцій для покращення асоціації треків. На відміну від попередніх підходів, ByteTrack не відкидає детекції з низькими значеннями впевненості, а навпаки – намагається використовувати їх з користю як додаткові потенційні об'єкти для асоціації.

Основні етапи роботи [12]:

- Детекція об'єктів – використовує детектор для отримання набору детекцій, які розділяються на два класи. Впевнені – з високим рівнем впевненості (вище заданого порогу), і невпевнені, у яких це значення між мінімально допустимим і порогом.
- Асоціація впевнених детекцій – використовує традиційні методи асоціації, такі як Угорський алгоритм та IoU, для співсчтавлення існуючих треків з впевненими детекціями.
- Асоціація невпевнених детекцій – переходить до невпевнених детекцій. Вони використовуються для оновлення тих треків, які не асоційовані на попередньому етапі. Це зберігає ті об'єкти, детекції яких на короткий час втратили значення впевненості, але все ще можуть бути ідентифіковані за слабкими ознаками.
- Актуалізація треків – оновлює координати об'єктів на основі асоціацій. Будь-які треки, які не були асоційовані протягом заданої кількості кадрів, видаляються назавжди.

ByteTrack є пристойним рішенням для задач трекінгу у складних умовах, гарно себе поводить при наявності перешкод або коли детекції слабкі чи неякісні, що робить його актуальним для інтеграції, до прикладу, в системи спостереження.

1.3.5 OC-SORT

OC-SORT розроблений як удосконалення класичного SORT знову ж для покращення стійкості треків у складних умовах, таких як оклюзія та нелінійний рух об’єктів. Відмінністю OC-SORT від SORT є те, що використовується підхід, орієнтований на спостереження (Observation-Centric), замість традиційного підходу, базованого на оцінці (Estimation-Centric).

Основні компоненти включають [13]:

- Буфер спостережень – зберігає останні спостереження об’єктів, щоб виконувати корекцію треків після періоду втрати детекцій. Це важливо у випадках оклюзій. Класичний фільтр Калмана в таких випадках сильно накопичує похибку прогнозу через відсутність спостережень.
- ORU – у разі повторної появи об’єкта після періоду відсутності використовує спостереження з буфера для корекції траєкторії замість того, щоб продовжувати траєкторію об’єкта за рахунок оцінок фільтру Калмана. Це дозволяє зменшити похибку, що накопичилася під час відсутності детекцій, а також асоціювати об’єкти з втраченими самими собою.

- OCM – щоб зберегти напрямок руху об’єкта після періоду відсутності детекцій, порівнює напрямок руху на момент останнього спостереження та нового виявлення об’єкта після оклюзії. Якщо напрямок руху суттєво різниться, то трек може бути скоригований для плавності траєкторії.

Послідовність алгоритму:

- Детекція об’єктів – об’єкти виявляються за допомогою детектора (YOLO, SSD).
- Асоціація треків – виконується за класичною схемою SORT із застосуванням Угорського алгоритму та IoU. Додатково використовує буфер спостережень.
- Корекція після оклюзії – після періоду відсутності детекцій використовується буфер спостережень для побудови віртуальної траєкторії об’єкта. Вона визначається за останнім спостереженням перед оклюзією та новим спостереженням після неї.
- Оновлення параметрів – після корекції фільтр Калмана оновлює свої параметри, використовуючи нові спостереження. Це дозволяє перейти до актуальних координат об’єкта.

Основними перевагами OC-SORT є висока ефективність у сценаріях із частими і тривалими оклюзіями та різкими змінами траєкторій об’єктів. Буфер спостережень дозволяє зменшити накопичення помилок і коригувати треки таким чином, щоб уникати розривів траєкторії та втрати об’єкту. Демонструє відносно хорошу швидкість, тому є варіантом для систем спостереження реального часу.

1.4 Модифікації фільтру Калмана

Класична версія обмежується лише лінійним рухом, що значно знижує її точність у випадках нелінійного. Для вирішення цієї проблеми існують дві модифікації – EKF і UKF, які дозволяють враховувати такі моменти.

EKF є розширеною версією звичайного фільтру Калмана, яка застосовується до нелінійних моделей. Основна відмінність полягає у використанні апроксимації лінійними моделями через застосування розкладу в ряд Тейлора першого порядку.

EKF складається з двох основних етапів:

- Прогнозування – визначення майбутнього стану об’єкта на основі попереднього стану, враховуючи нелінійні моделі руху.

- Оновлення – корекція прогнозу на основі нових спостережень. На цьому етапі використовується як лінійна апроксимація, так і обчислення часткових похідних для оцінки змінних стану.

Є корисним у тих випадках, коли модель руху об’єктів може бути апроксимована лінійними функціями, але у складних нелінійних системах буде мати обмеження.

UKF розроблено для подолання цих обмежень, пов’язаних з лінійною апроксимацією. На відміну від EKF не використовує розклад у ряд Тейлора. Замість цього застосовує трансформацію безлінійного наближення (Unscented Transform) для побудови сигма-точок, що забезпечує більш точну оцінку стану у випадках сильної нелінійності.

Основні етапи включають:

- Формування сигма-точок – замість апроксимації формує набір сигма-точок, які відображають розподіл стану. Ці точки дозволяють отримати набір можливих положень об’єкта на наступному кроці.
- Прогнозування – сигма-точки пропускаються через нелінійну модель, що дозволяє отримати прогноз нового стану.
- Оновлення – сигма-точки використовуються для оцінки коваріаційної матриці та відповідно оновлення стану на основі вже нових спостережень.

EKF використовується для відстеження об’єктів з прогнозованим лінійним рухом. У той же час UKF краще себе поводить у випадках нелінійного руху, наприклад, для відстеження дронів, пішоходів або інших об’єктів з непередбачуваними траєкторіями, але має вищу часову складність [14],[15].

1.5 Висновки до розділу 1

У першому розділі роботи було розглянуто основні методи виявлення та відстеження об’єктів, які є основою для подальшої реалізації. Детально проаналізовано сучасні підходи до виявлення об’єктів, зокрема методи диференціювання кадрів, оптичного потоку, відокремлення фону, а також детектори з використанням нейромереж – YOLO, SSD та прискорені R-CNN.

Традиційні методи на основі аналізу руху мають більшу обчислювальну швидкість, проте значно поступаються нейромережевим детекторам за точністю та логічністю виявлення об’єктів за складних умов, таких як зміни освітлення або часткові оклюзії. Серед нейромережевих детекторів, YOLO вирізняється більшою

швидкістю, SSD – балансом між швидкістю та точністю, а прискорені R-CNN забезпечують найвищу точність за рахунок використання двоетапної архітектури та часу обчислення.

У контексті трекінгу об’єктів було розглянуто основні алгоритми, зокрема фільтр Калмана, частинковий фільтр, DeepSORT, ByteTrack і OC-SORT. Фільтр Калмана досить сильно чутливий до шуму та нестійкий до нелінійних змін, у зв’язку з чим неефективний у складних умовах. Його покращені версії EKF і UKF дозволяють враховувати нелінійність руху об’єктів шляхом застосування апроксимаційних методів. Зокрема, EKF забезщпечує певну адаптивність до нелінійних змін, але при цьому підвищує обчислювальну складність алгоритму. UKF демонструє вищу стабільність та точність у сценаріях із сильними нелінійностями та непередбачуваними змінами руху об’єктів ціною швидкості.

DeepSORT та ByteTrack покращують базовий фільтр Калмана завдяки використанню ReID та інтеграції невпевнених детекцій відповідно. Обидва алгоритми показують гарні результати у порівнянні з базовим, але залишаються вразливими до тривалих оклюзій, що може призводити до розриву траєкторій, що ускладнює аналіз повних маршрутів об’єктів протягом появи в кадрі.

OC-SORT орієнтований на усунення накопичення помилок під час тривалих оклюзій шляхом використання буфера спостережень та механізму корекції траєкторій, і який дозволяє отримати пристойні результати щодо відстеження повних траєкторій об’єктів у кадрі, навіть при довгих зникненнях.

РОЗДІЛ 2
АНАЛІЗ АЛГОРИТМІВ ВІДСТЕЖЕННЯ ОБ'ЄКТІВ

2.1 Проблеми трекінгу в умовах оклюзії та нелінійного руху

Під час відстеження об'єктів на відео однією з ключових проблем є оклюзія. Вона відбувається тоді, коли об'єкт частково або повністю перекривається іншим об'єктом або виходить за межі кадру. Це призводить до втрати детекцій та порушення безперервності траєкторії об'єкта.

Сучасні алгоритми трекінгу прогнозують нове положення об'єкта на основі попередніх. Проте в умовах тривалої оклюзії або різких змін траєкторії ці методи можуть поводити себе неочікувано. Трекер може або втратити об'єкт, або помилково призначити ідентифікатор іншому об'єкту, що спричиняє проблему змішування та перепризначення ідентифікаторів.

Нелінійний рух об'єктів також є проблемою. У реальних умовах об'єкти можуть різко змінювати траєкторію, прискорюватись або сповільнюватись, змінювати масштаб. Це порушує припущення про лінійний рух, на якому базуються більшість трекерів, зокрема й фільтр Калмана. У таких випадках алгоритми, що використовують лінійні моделі, накопичують помилки, що призводить до зміщення траєкторій.

Прикладами ситуацій, коли ці проблеми найбільш помітні:

- Густі сцени з великою кількістю рухомих об'єктів (MOT20, DanceTrack)
- Різкі зміни напрямку руху об'єктів
- Велика кількість перехрещень траєкторій об'єктів у сцені, що призводить до частих зіткнень та оклюзій

Для подолання цих проблем використовуються різні підходи, зокрема:

- Буфер спостережень для збереження інформації про останні відомі координати об'єкта до моменту його зникнення.
- Метрики відповідності, такі як відстань Махаланобіса [16], для врахування форми та розміру об'єкта під час асоціації треків.
- Механізми повторної асоціації треків після оклюзії.

2.2 Методи асоціації треків

Від правильності асоціації залежить, чи зможе алгоритм коректно підтримувати ідентифікатор об’єкта протягом усього руху, зберігаючи його цілісність та коректну траєкторію.

2.2.1 Метод IoU

Метод IoU базується на обчисленні площі перетину та об’єднання двох обмежувальних рамок (bounding boxes). Формула визначається як:

$$IoU = \frac{Area(BB_{pred} \cap BB_{real})}{Area(BB_{pred} \cup BB_{real})}$$

де BB_{pred} – спрогнозована рамка об’єкта,

BB_{real} – реальна рамка.

Після цього дану метрику порівнюють з певним порогом, і якщо вона менше нього, відповідно об’єкт вважається втраченим в конкретному моменті, і навпаки.

Є швидким і простим у реалізації, проте іноді вразливий до змін масштабу та форми об’єктів, великих швидкостей, різких змін напрямку руху, розташування об’єктів близько один до одного.

2.2.2 Угорський алгоритм

Застосовується для оптимального вирішення задачі призначення, а не схожості, як IoU. Суть полягає у визначенні найкращої відповідності між наборами детекцій та спрогнлозованих рамок.

Основні переваги:

- Завжди знаходить оптимальне рішення для задачі призначення.
- Стійкість до зміни кількості об’єктів на кадрі.
- Може обробляти велику кількість треків одночасно.

Головним недоліком алгоритму є часова складність $O(n^3)$, що може негативно впливати на швидкодію програми.

2.3 Використання метрик відповідності

Під час асоціації необхідний метод оцінки схожості або відстані між поточними детекціями та вже існуючими треками. Для цього використовуються метрики відповідності, які оцінюють такі різниці між об’єктами на поточному та попередньому кадрах. Серед основних метрик можна виділити вже розглянуту IoU, а також нормалізовану відстань Васерштейна та відстань Махаланобіса.

2.3.1 Нормалізована відстань Васерштейна

Є узагальненою метрикою для оцінки схожості між розподілами. Використовується для вимірювання відстані між двома розподілами шляхом обчислення мінімальної вартості переміщення маси з одного розподілу до іншого.

У контексті відповідності детекцій та спрогнозованих рамок оцінює, наскільки схожі розташування та розміри двох об’єктів, розглядаючи їх як розподіли маси. Визначає, наскільки потрібно перемістити один об’єкт, щоб він співпав з іншим.

Переваги:

- Можливість враховувати різні параметри об’єктів (форма, розмір, колір).
- Чутливість до змін форми об’єктів.
- Менша ймовірність хибних спрацьовувань у порівнянні з IoU.

Недоліком є більша обчислювальна складність.

2.3.2 Відстань Махаланобіса

Є статистичною метрикою, яка вимірює відстань між точкою та розподілом шляхом урахування коваріаційної матриці.

В контексті є корисною для асоціації треків, коли об’єкти мають певні корельовані ознаки, бо враховує кореляції між різними компонентами об’єкта, є більш надійним для мультиоб’єктного трекінгу. Аналогічно до NWD, є відносно обчислювально затратною метрикою у реальному часі [16].

2.4 Фактори, що впливають на стабільність треків

Стабільність треків у системах відстеження залежить від багатьох факторів, серед яких якість детекцій, налаштування трекера та менеджмент ідентифікаторів.

Якість детекцій відіграє важливу роль у цілісності траєкторій. Високоточні детекції дозволяють підтримувати стабільні треки, тоді як відсутність або хибні детекції призводять до втрати об’єктів або помилкового перепризначення або перемішування ідентифікаторів.

Параметри, такі як мінімальний вік об’єкта (min age), максимальний термін відсутності (max missed) та поріг для IoU, безпосередньо впливають на те, як довго трек зберігається після втрати детекцій та наскільки суворо відбувається асоціація об'єктів.

Менеджмент ідентифікаторів визначає правила присвоєння та збереження ID об’єктів. У випадках, коли об’єкт тимчасово зникає з кадру, коректне збереження його ідентифікатора є важливим для відновлення траєкторії після повторної появи. У DeepSORT та OC-SORT це завдання вирішується за допомогою ReID і буферу спостережень відповідно, які коректують треки після оклюзій [11],[13].

2.5 Висновки до розділу 2

У розділі було проведено аналіз алгоритмів трекінгу об’єктів з акцентом на проблеми, що виникають за умов оклюзії і нелінійного руху. Було розглянуто різні методи асоціації треків, такі як IoU й Угорський алгоритм, які використовуються для збереження цілісності траєкторій під час трекінгу.

Були проаналізовані метрики відповідності (IoU, NWD, відстань Махаланобіса) на точність асоціації об’єктів, зокрема в умовах зміни масштабу та частих перетинів траєкторій.

Окремо розглянуто вплив різних факторів на стабільність треків. Вони визначають баланс між швидкістю реагування трекера на нові детекції та здатністю зберігати треки при тимчасових оклюзіях. Оптимальне налаштування цих параметрів підвищує стабільність роботи трекера.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМУ OC-SORT

3.1 Вибір методології

З перших двох розділів основними проблемами алгоритмів для відстеження об’єктів на відео є нелінійний рух об’єктів, який важко передбачати банальними шляхами, а також накопичення похибки і відхилення траєкторій через тривалі оклюзії. Метою роботи стала розробка та реалізація розглянутого вище алгоритму OC-SORT з кількох причин:

- Фокус на вище описаних проблемах

  Дозволяє коректувати траєкторії після зникнення об’єктів, тобто має покращитись ситуація з тривалими оклюзіями.

- Актуальність та новизна методу

  Метод був запропонований відносно нещодавно і на момент реалізації не знайдений в загально відомих бібліотеках, таких як OpenCV чи TensorFlow, що робить його недоступним для інтеграції в проєкти. Це робить його реалізацію актуальною для практичного застосування.

- Можливість модифікацій/адаптацій

  Є можливість адаптувати компоненти під конкретні вимоги, зокрема це зміна параметрів буфера спостереження, формул оновлення треків, метрик для асоціації.

Тобто реалізація OC-SORT трекера вирішує поставлені задачі.

3.2 Архітектура алгоритму

Іде перехід від оціночного підходу (як у SORT) до спостережувального підходу. Це зменшує похибку у випадках втрат об’єктів чи при складному русі. Основні етапи наступні [13]:

- Передбачення стану

  Використання фільтру Калмана для прогнозування місцезнаходжень об’єктів в наступний момент часу. Це відбувається на основі попереднього місцезнаходження та швидкості:

$$x_{t|t-1} = Fx_{t-1|t-1}$$
$$P_{t|t-1} = FP_{t-1|t-1}F^T + Q_t$$

де

$x_{t|t-1}$ – прогнозований стан на наступному кроці,

F – матриця переходу станів,

$P_{t|t-1}$ – коваріаційна матриця стану,

Q_t – матриця шуму процесу.

- Оновлення стану

  В моменти, коли об’єкти видимі, наявний прогноз комбінується з наявними спостереженями, щоб компенсувати похибки прогнозу, якщо рух нелінійно змінюється:

$$K_t = P_{t|t-1}H^T(HP_{t|t-1}H^T + R_t)^{-1}$$
$$x_{t|t} = x_{t|t-1} + K_t(z_t - Hx_{t|t-1})$$
$$P_{t|t} = (I - K_tH)P_{t|t-1}$$

де

K_t – коефіцієнт Калмана,

z_t – нове спостереження,

H – матриця спостереження,

R_t – коваріаційна матриця шуму спостереження.

- Асоціація треків

  Передбачені рамки з спостереженнями асоціюються з використанням метрики IoU для визначення схожості між ними. Додатково використовується метрика, що дивиться за узгодженістю напрямку руху об’єктів.

- Підтримка треків

  Протягом всього процесу іде управління існуючими треками, як активними, так і тимчасово втраченими. Якщо втрачений трек є таким вже більше певного порогу кількості кадрів, то він стає повністю втрачним і видаляється.

3.3 Буфер спостережень

Роль буфера у збереженні останніх спостережень за об’єктом, щоб використати потім ці дані для асоціації та відновлення треку з тим самим ідентифікатором у майбутньому після оклюзій.

У проєкті має реалізацію як структура даних з обмеженою кількістю треків, а саме як черга, що зберігає останні координати об’єкти у вигляді [x, y, w, h], де

x, y – координати центру об’єкта,

w, h – ширина та висота рамок навколо об’єкта.

На кожен об’єкт в черзі виділяється n місць під спостереження, нові спостереження додаються до буфера, найстаріщі автоматично видаляються, таким чином запобігаючи переповненню пам’яті. Буфер також виористовується для оновлення фільтру Калмана після оклюзій.

3.4 Оновлення фільтру Калмана при оклюзії (ORU)

ORU коригує траєкторії об’єктів після їх повторної детекції після зникнення. Дані, накопичені у буфері спостережень, використовуються для корекції параметрів фільтру Калмана цього об’єкта. Завдяки його використанню виходить покращення якості траєкторії порівняно з базовими алгоритмами з фільтром Калмана. Реалізація в межах класів Track і OCTracker.

Екземпляри класу Track – це треки, кожен з яких відповідає одному об’єкту (рис. 3.1):

```
class Track:
    def __init__(self, track_id, bbox):
        self.id = track_id
        self.kf = KalmanFilter(bbox)
        self.buffer = ObservationBuffer()
        self.buffer.add(bbox)
        self.age = 1
        self.missed = 0
        self.active = True
```

Рис. 3.1
Клас Track

Де:

- id – ідентифікатор треку (об’єкту в ідеальному сценарії)
- kf – фільтр Калмана для прогнозування руху,
- buffer – буфер спостережень об’єкта.
- age – к-ть життя треку (в кадрах)
- missed – к-ть кадрів з моменту втрачання об’єкта
- active – об’єкт дійсний/втрачений

Клас OCTracker – основний клас, який пов’язує всю логіку та реалізує процес ORU в методі update() (рис. 3.2):

- Виконується прогноз усіх дійсних треків (track.predict()).
- Виконується асоціація нових детекцій з отриманими прогнозами на основі метрики IoU (рис 3.3).
- Якщо об’єкт став відсутній (unmatched_tracks), ініціюється mark_missed(). Усі об’єкти, у тому числі втрачені, при виявленні (matches) ініціюють оновлення.
- Виконується корекція параметрів фільтру Калмана

```python
class OCTracker:
    def __init__(self, iou_threshold=0.3, max_missed=30):
        self.tracks = []
        self.next_id = 1
        self.iou_threshold = iou_threshold
        self.max_missed = max_missed

    def update(self, detections):
        for track in self.tracks:
            track.predict()

        predicted_boxes = [track.get_state() for track in self.tracks]
        matches, unmatched_detections, unmatched_tracks = associate(detections, predicted_boxes, self.iou_threshold)

        for det_idx, trk_idx in matches:
            self.tracks[trk_idx].update(detections[det_idx])

        for idx in unmatched_detections:
            new_track = Track(None, detections[idx])
            self.tracks.append(new_track)

        for idx in unmatched_tracks:
            self.tracks[idx].mark_missed()

        self.tracks = [t for t in self.tracks if t.missed <= self.max_missed]

        outputs = []
        min_age = 3

        for t in self.tracks:
            if t.active:
                if t.id is None and t.age >= min_age:
                    t.id = self.next_id
                    self.next_id += 1

                if t.id is not None:
                    x, y, w, h = t.get_state()
                    outputs.append((int(x), int(y), int(w), int(h), t.id))
```

Рис. 3.2
Клас OCTracker

```python
def associate(detections, trackers, iou_threshold=0.3):
    if len(trackers) == 0:
        return [], list(range(len(detections))), []

    iou_matrix = np.zeros((len(detections), len(trackers)), dtype=np.float32)

    for d, det in enumerate(detections):
        for t, trk in enumerate(trackers):
            iou_matrix[d, t] = compute_iou(det, trk)

    matched_indices = linear_sum_assignment(-iou_matrix)
    matched_indices = np.asarray(matched_indices).T

    matches = []
    unmatched_detections = list(range(len(detections)))
    unmatched_tracks = list(range(len(trackers)))

    for d, t in matched_indices:
        if iou_matrix[d, t] >= iou_threshold:
            matches.append((d, t))
            unmatched_detections.remove(d)
            unmatched_tracks.remove(t)

    return matches, unmatched_detections, unmatched_tracks
```

Рис. 3.3
Асоціація по IoU

3.5 Структура проєкту

У межах проєкту з ціллю не тільки розробити один із нових методів трекінгу, а й зробити пристойне тестування та порівняльний аналіз результату з чимось іншим, було реалізовано два різні детектори та три різні трекери (OC-SORT включно). Додатково створений файл для утиліт, де реалізована метрика IoU, а також інструменти для завантаження набору кадрів з відео файлу в оперативну пам’ять. Програму є можливість запускати окремо як під відеофайли, так і під відеокамеру на комп’ютері.

Детектори (рис. 3.4):

- Диференціювання кадрів (frame differencing) – алгоритмічний детектор, який базується на аналізі різниці між двома послідовними кадрами. Кадри перетворюються на відтінки сірого, застосовується Гауссове розмивання, щоб прибрати різкі шуми, після чого береться різниця по пікселях та шукаються контури абсолютно всіх вагомих змін, які вважаються рухами.
- YOLO – нейромережевий детектор на базі моделі YOLOv8, попередньо навчений на 79 класах об’єктів і налащтований під поставлені задачі.

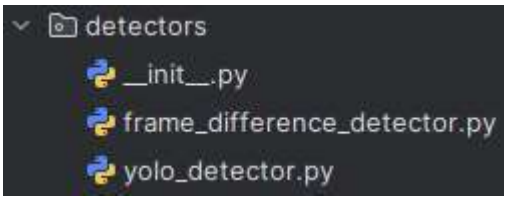

Рис. 3.4
Детектори

Трекери (рис. 3.5):

- Трекер на детекціях (detection simple tracker) – перший і найпростіший з трьох трекер, який використовує у своїй логіці лише певний тип детектора, і на кожному кадрі відмальовує отримані детекції. Таким чином, у даному випадку немає навіть ідентифікаторів об’єктів. Корисний для порівнянь як базовий рівень.
- Трекер на фільтрі Калмана (Kalman simple tracker) – другий трекер, який окрім детекцій зроблений на базі фільтру Калмана та SORT алгоритму, використовуючи для асоціацій метрику IoU. У даному випадку трекер вже видає рамки об’єктів разом з ідентифікаторами, маючи вже певні

інструменти для запобігання оклюзії та менеджмента цих ідентифікаторів між об’єктами.

- Трекер OC-SORT (Kalman OC-SORT tracker) – третій і основний трекер, який було розглянуто і описано вище, який є основним результатом роботи.

Рис. 3.5
Трекери

3.6 Тестування алгоритму

Для запуску однієї конфігурації потрібно один з двох детекторів, один з трьох трекерів і один з двох джерел відео. Отже сценаріїв запуску програми є дванадцять, кожен з яких було протестовано. Для тестування та порівняння шести сценаріїв на відеофайлі було обрано один спільний запис, зроблений спеціально під відповідні задачі. Результатом тестування програми на відеофайлі є зафіксовані дві метрики:

- Кардри на секунду (Frames per second / FPS) – кількість кадрів в секунду, яку видає програма в середньому протягом циклу тестування.

- К-ть ID (Track IDs total) – число, яке вказує на те, скільки всього ідентифікаторів було присвоєно об’єктам, що рухались протягом циклу тестування. Чим це число є більше, тим гірше спрацював алгоритм, бо це значить, що на однаковому відеофайлі з однаковою кількістю об’єктів він видав більше унікальних ідентифікаторів, отже більше виражена проблема переприсвоєнь.

Результатом тестування програми на відеокамері є лише середнє значення FPS, а також висновок щодо працездатності її коректно відслідковувати великі об’єкти, що займають майже весь простір кадру. Для тестування використовувалась камера в 30 FPS.

Таблиця 3.1
Відеофайл, детектор диференціювання кадрів

Трекер	FPS (обмежений)	FPS	Track IDs total
Трекер на детекціях	65	1873	Немає
Трекер на фільтрі Калмана (рис. 3.6)	64	1260	265
Трекер OC-SORT	64	1230	134

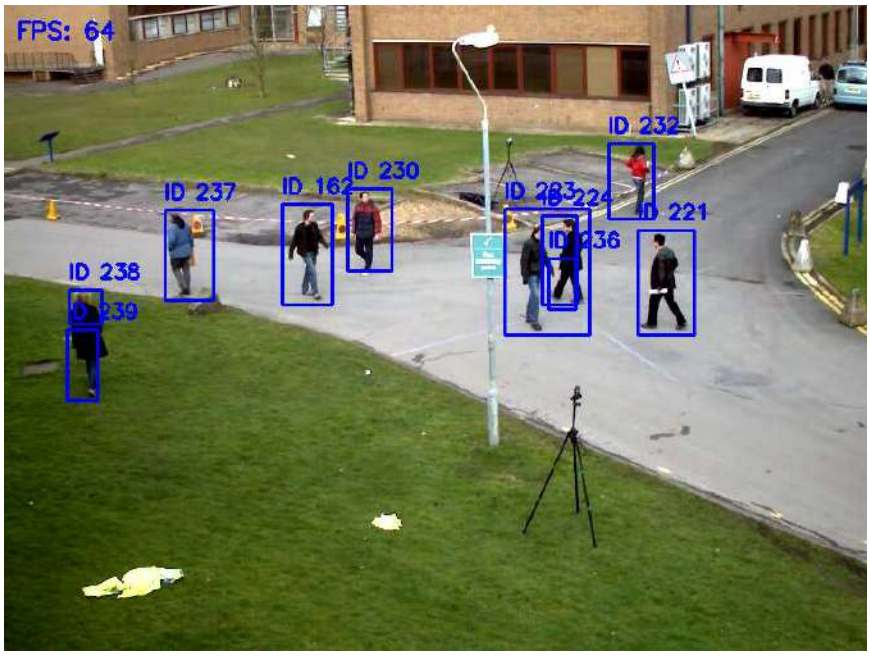

Рис. 3.6
Відео, детектор диференціювання кадрів, трекер на фільтрі Калмана
Протягом відео налічено 265 ідентифікаторів

Таблиця 3.2
Відеофайл, YOLO детектор

Трекер	FPS	Track IDs total
Трекер на детекціях	31	Немає
Трекер на фільтрі Калмана	31	84
Трекер OC-SORT (рис. 3.7)	31	54

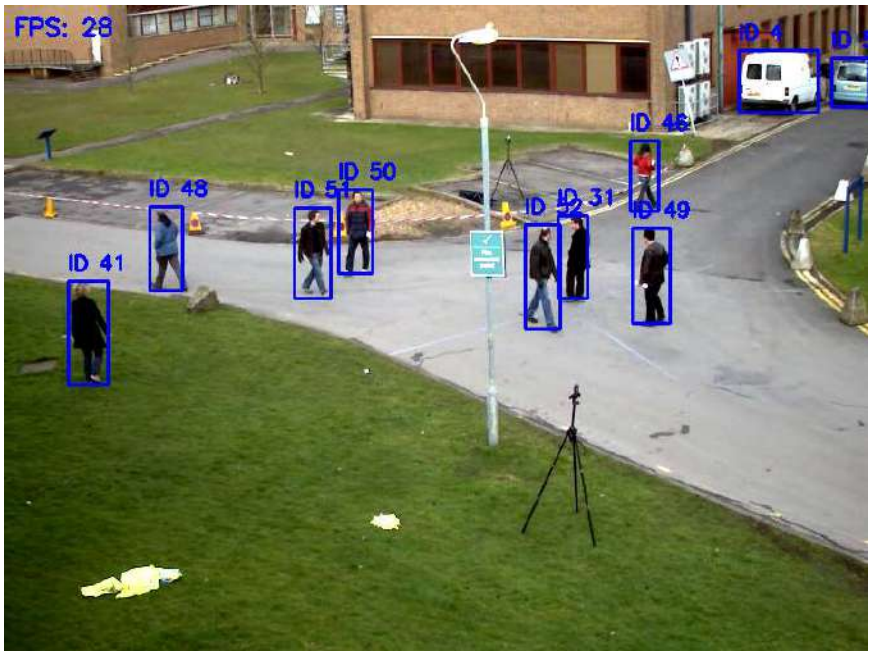

Рис. 3.7
Відео, YOLO детектор, трекер OC-SORT
Протягом відео налічено 54 ідентифікатора

Таблиця 3.3
Відеокамера, детектор диференціювання кадрів

Трекер	FPS	Коректно працює на великих об’єктах
Трекер на детекціях	27	Ні
Трекер на фільтрі Калмана	26	Ні

Трекер OC-SORT (рис. 3.8)	26	Ні

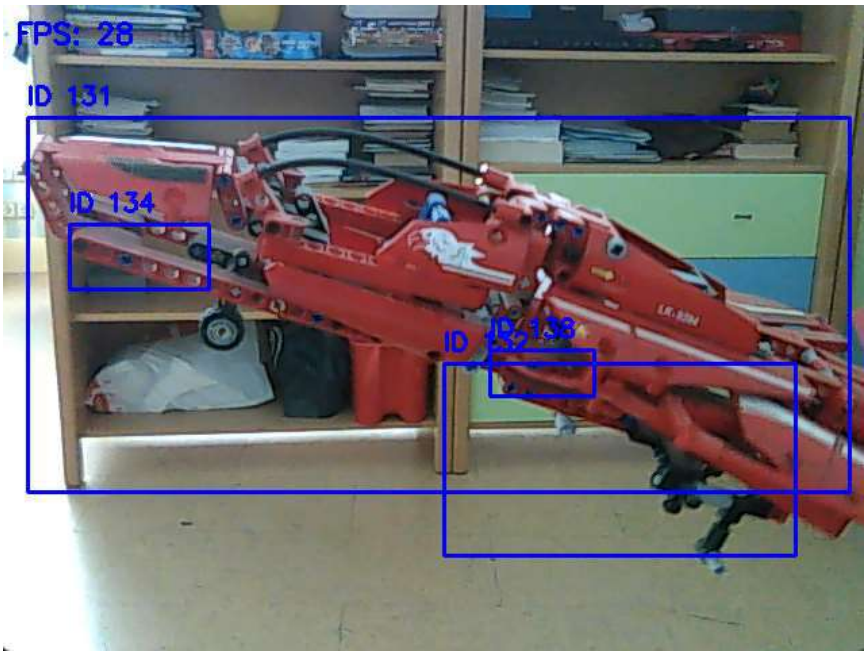

Рис. 3.8
Камера, детектор диференціювання кадрів, трекер OC-SORT
Робота на великому об'єкті

Таблиця 3.4
Відеокамера, YOLO детектор

Трекер	FPS	Коректно працює на великих об'єктах
Трекер на детекціях	18	Так
Трекер на фільтрі Калмана	18	Так
Трекер OC-SORT (рис. 3.9)	18	Так

Рис. 3.9
Камера, YOLO детектор, трекер OC-SORT
Робота на великому об'єкті

Окрім базових тестувань, було проведено експеримент з налаштуваннями трекерів на базі фільтру Калмана. Об'єктом тестування був параметр мінімальний вік об'єкту (*min age threshold)*. Під час тестування було помічено, що трекери з фільтром Калмана під час роботи могли часто помічати в кадрі якийсь мінімальний рух пікселів на статичному малому об'єкті і зареєструвати їх як унікальні об'єкти на буквально декілька кадрів, після чого вони зникали, і таким чином в цикл ідентифікаторів додається плюс один. Це було причиною надмірного зростання метрики *Track IDs total*, через що було створено параметр *min age threshold*, який відповідає за мінімальну кількість кадрів, які об'єкт повинен бути в полі зору програми, щоб його зареєструвало.

Таблиця 3.5
Результати експерименту на трекері OC-SORT, Відеофайл

Трекер OC-SORT	Без *min age threshold*	З *min age threshold*
Track IDs total (детектор диференціювання кадрів)	235	134
Track IDs total (YOLO детектор)	79	54

Усі тестування трекеру на фільтрі Калмана і трекеру OC-SORT проводились з уже доданим параметром *min age threshold*.

3.7 Аналіз результатів тестування

За результатами тестування на відеофайлі та простішому детекторі диференціювання кадрів можна стверджувати, що по швидкодії найпростіший трекер видає найкращий результат в 1873 кадри на секунду, що не є дивним, адже логіки трекінгу як такої немає. Простий трекер і основний трекер є повільнішими по швидкодії, але показник у 1260 і 1230 кадрів на секунду це і там є неймовірно високим показником. Важливо те, що по швидкодії основний OC-SORT трекер втрачає відносно другого трекера лише 2.4% швидкодії, що є чудовим показником, враховуючи те, що кількість ідентифікаторів протягом відео знизилася на 49.4%. Це каже про те, що реалізований основний трекер покращує свою минулу версію майже в 2 рази, майже не жертвуючи швидкодією.

З тим самим відеофайлом, але детектором YOLO, по-перше, усі трекери видали швидкодію в 31 кадр на секунду, що говорить про значне збільшення обчислень під час роботи, проте 30 кадрів на секунду це те значення, яке абсолютно припустиме в системах реального часу. Але зі збільшенням якості детектора, можна побачити, що відносно простішого, трекер на фільтрі Калмана видав 84 ідентифікатора, що є на 68.3% або в 3 рази менше, що є неймовірним результатом. Аналогічно, трекер OC-SORT видав 54 ідентифікатора, що на 59.7% менше, ніж при простішому детекторі та на 35.7% кращим показником, ніж у трекера на фільтрі Калмана.

За результатами тестувань на камері підтверджується те, що з детектором YOLO затрати на обчислення сильно збільшуються, простий детектор в середньому видає 26-27 кадрів із 30, які видає сама камера, у той час на YOLO це 18 кадрів на секунду. Проте з наглядного прикладу видно, що алгоритм детектору, який спирається на математичну різницю двох кадрів за пікселями, не може впоратися в ситуаціях, коли на кадрах великі детальні об’єкти, розділяючи їх на багато менших частин, вважаючи, що об’єктів насправді декілька. У той час нейромережевий детектор працює коректно і може впоратися з даною задачею ціною швидкодії.

Останній експеримент з параметром мінімального порогу кадрів, які об’єкти повинні бути на кадрі для помічення їх, показав, що така мала деталь може серйозно покращити результат, не враховуючи випадкові миттєві детекції і не призначуючи їм ідентифікатори. Такми чином для детекторів результат на трекері

OC-SORT покращився на 43.0% і 31.6% відповідно, нічим при цьому не жертвуючи.

3.8 Висновки до розділу 3

Цей розділ був присвячений розбору вибраного для реалізації методу трекінгу OC-SORT. Вибір був обгрунтований перевагами даного алгоритму, зокрема його здатністю до зменшення похибки в моменти, коли об’єкт зникає з поля видимості на кадрі, шляхом створення буфера спостережень за об’єктами, а також механізму ORU, який коригує траєкторії об’єктів і параметри фільтру Калмана після виявлення їх заново.

Було описано архітектуру алгоритму, які наявні етапи та що відбувається під час кожного з них. Продемонстровано те, як окремі компоненти алгоритму були реалізовані та зв’язані між собою в програмі.

Розглянуто структуру проєкту, де окрім основного трекеру, було реалізовано ще два детектори і два інших трекери для проведення тестувань і порівнянь.

Остання та основна частина заключалася в проведенні повноцінного тестування всіх можливостей програми шляхом перебору всіх сценаріїв і фіксації потрібних для порівняння та висновків метрик *FPS* і *Track IDs total*. Результати тестування було проаналізовано та зроблено висновки, які співпадають з припущеннями, на яких будувалася основна задача роботи, і реалізований алгоритм OC-SORT показав гарні результати. Додатково до цього експериментально отримано підтвердження того, що підвищення якості детектора може спокійно давати відповідні покращення в стабільності роботи.

ВИСНОВКИ

У цій кваліфікаційній роботі було проведено комплексне дослідження методів виявлення та відстеження об’єктів у відеопотоці з акцентом на підвищення стабільності трекінгу у складних умовах, таких як оклюзія та нелінійний рух. Результатом роботи стала реалізація алгоритму OC-SORT, що поєднує методи обробки даних з буфером спостережень та механізмами корекції траєкторій після тимчасових зникнень об’єктів.

В основній частині було розглянуто теоретичні засади відстеження об’єктів у відеопотоці, зокрема основні методи виявлення об’єктів, традиційні та нейромережеві, а також сучасні алгоритми трекінгу, більшість з яких працюють із використанням фільтру Калмана. Було проведено аналіз цих алгоритмів, зокрема визначено наявні проблеми та можливі підходи до їх вирішення. Обгрунтовано вибір конкретного методу для реалізації, проведено його розбір, демонстрація, і тестування, включаючи декілька інших для порівняння, після чого проаналізовано отримані результати та зроблено висновки.

Отже, результатом роботи є реалізація трекеру з впровадженим механізмом спостереження за об’єктами, що дозволяє коригувати і покращувати їх траєкторії під час трекінгу, покращуючи роботу програми в складних умовах таких як оклюзія та нелінійний рух.

Реалізація може бути використана для створення систем реального часу для спостереження за певними об’єктами у складних умовах, до прикладу пішоходами або транспортними засобами.

Для подальших досліджень є можливості додаткової оптимізації та покращення алгоритму шляхом додавання існуючих або нових метрик відповідності, інтеграція в програму додаткових нейронних мереж з метою роботи над певними додатковими функціями, розробка автоматичної стратегії налаштування та адаптації параметрів трекера в реальному часі для ще більшого покращення продуктивності.

Таким чином, результат роботи та тестувань підтверджує ефективність розглянутої методології OC-SORT для відстеження рухомих об’єктів на відео. Впровадження описаних механізмів спостережень та корекції виправдало очікування та покращило результати в порівнянні з іншими алгоритмами на практичному прикладі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dai Y., Hu Z., Zhang S. and Liu L., “A survey of detection-based video multi-object tracking,” Displays, vol.75, pp.102317, 2022. https://doi.org/10.1016/j.displa.2022.102317.
2. Bashar M.K., Islam S., Hussain K.K., Hasan M.B., Rahman A.A.B.M. and Kabir M.H., “Multiple Object Tracking in Recent Times: A Literature Review,” arXiv preprint arXiv:2209.04796, 2022. https://doi.org/10.48550/arXiv.2209.04796.
3. Parger M., Tang C., Neff T., Twigg C.D., Keskin C., Wang R. and Steinberger M., “MotionDeltaCNN: Sparse CNN Inference of Frame Differences in Moving Camera Videos,” arXiv preprint arXiv:2210.09887, 2022. https://doi.org/10.48550/arXiv.2210.09887.
4. Alfarano A., Maiano L., Papa L. and Amerini I., “Estimating optical flow: A comprehensive review of the state of the art,” Computer Vision and Image Understanding, vol.249, pp.104160, 2024. https://doi.org/10.1016/j.cviu.2024.104160.
5. Bouttefroy P.L.M., Bouzerdoum A., Phung S.L. and Beghdadi A., “On the analysis of background subtraction techniques using Gaussian Mixture Models,” IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, USA, pp. 4042-4045, 2010. https://doi.org/10.1109/ICASSP.2010.5495760.
6. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y. and Berg A.C., “SSD: Single Shot MultiBox Detector,” arXiv preprint arXiv:1512.02325, 2016. https://doi.org/10.48550/arXiv.1512.02325.
7. Varghese R. and Sambath M., “YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness,” 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, pp. 1-6, 2024. https://doi.org/10.1109/ADICS58448.2024.10533619.
8. Ren S., He K., Girshick R. and Sun J., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” arXiv preprint arXiv:1506.01497, 2016. https://doi.org/10.48550/arXiv.1506.01497.
9. Pei Y., Biswas S., Fussell D.S. and Pingali K., “An Elementary Introduction to Kalman Filtering,” arXiv preprint arXiv:1710.04055, 2019. https://doi.org/10.48550/arXiv.1710.04055.
10. Zhao Q., Dong L., Chu X., Liu M., Kong L., and Zhao Y., “Particle Filter Tracking System Based on Digital Zoom and Regional Image Measure,” Sensors 25, no. 3: 880, 2025. https://doi.org/10.3390/s25030880.
11. Perera I. et al., “Vehicle Tracking based on an Improved DeepSORT Algorithm and the YOLOv4 Framework,” 2021 10th International Conference on

Information and Automation for Sustainability (ICIAfS), Negambo, Sri Lanka, pp. 305-309, 2021. https://doi.org/10.1109/ICIAfS52090.2021.9606052.

12. Zhang Y., Sun P., Jiang Y., Yu D., Weng F., Yuan Z., Luo P., Liu W. and Wang X., "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," arXiv preprint arXiv:2110.06864, 2022. https://doi.org/10.48550/arXiv.2110.06864.
13. Cao J., Pang J., Weng X., Khirodkar R. and Kitani K., "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking," arXiv preprint arXiv:2203.14360, 2023. https://doi.org/10.48550/arXiv.2203.14360.
14. Li Q., Li R., Ji K. and Dai W., "Kalman Filter and Its Application," *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Tianjin, China, pp. 74-77, 2015. https://doi.org/10.1109/ICINIS.2015.35.
15. Montella C., "The Kalman Filter and Related Algorithms," ResearchGate, 2011. https://www.researchgate.net/publication/236897001_The_Kalman_Filter_and_Related_Algorithms_A_Literature_Review.
16. Pinho R.R., Tavares J.M.R.S. and Correia M.V., "Efficient Approximation of the Mahalanobis Distance for Tracking with the Kalman Filter," ResearchGate, 2007. http://dx.doi.org/10.2507/IJSIMM06(2)S.03.