МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

СІДЛЕЦЬКИЙ Тарас Андрійович

Програмний модуль виявлення та відстеження рухомих об'єктів у відеопотоці на вбудованій обчислювальній платформі BeagleBone AI / Software Module for Detection and Tracking of Moving Objects in Video Streams on the Embedded BeagleBone AI Platform

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
Сідлецький Т.А.

_____

Науковий керівник
к.т.н., доцент Д.І. Загородня

_____

Кваліфікаційну роботу допущено до захисту
«___»_____ 20___ р.
Завідувач кафедри
_____ Н. В. Дзюбановська

Тернопіль – 2026

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.В. Дзюбановська
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
СІДЛЕЦЬКИЙ Тарас Андрійович
(прізвище, ім'я, по батькові)

1. Програмний модуль виявлення та відстеження рухомих об'єктів у відеопотоці на вбудованій обчислювальній платформі BeagleBone AI / Software Module for Detection and Tracking of Moving Objects in Video Streams on the Embedded BeagleBone AI Platform
керівник роботи Д. І. Загородня к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 01 грудня 2025 р. № 894.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2026 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна документація платформи BeagleBone AI-64, документація бібліотек комп'ютерного зору та машинного навчання, експериментальні дані для тестування програмного модуля.
4. Основні питання, які потрібно розробити:
- аналіз сучасних методів виявлення та відстеження рухомих об'єктів у відеопотоці;
- аналіз архітектурних особливостей і можливостей платформи BeagleBone AI-64;
- розробка програмного модуля виявлення та відстеження об'єктів на основі алгоритмів YOLOv8 і D□□□S□RT;
- програмна реалізація та оптимізація програмного модуля з використанням апаратних прискорювачів;
- експериментальна перевірка точності виявлення, якості відстеження та продуктивності системи.

5. Перелік графічного матеріалу в роботі:
- конвеєр захоплення відеоданих та детекції об'єктів;
- конвеєр відстеження об'єктів (DeepSORT) та візуалізації;
- функціональна схема програмного модуля;
- структурна схема системи на базі SoC TDA4VM;
- діаграма потоків даних програмного конвеєра;
- схема асоціації даних та оновлення життєвого циклу треків.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 1 грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2026 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2026 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 15.03.2026 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2026 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2026 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2026 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2026 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2026 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2026 р.	

Студент_____ Т.А. Сідлецький
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Д. І. Загородня
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль виявлення та відстеження рухомих об’єктів у відеопотоці на вбудованій обчислювальній платформі BeagleBone AI» на здобуття освітнього ступеня «Бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом 74 сторінки і містить 18 ілюстрацій, 8 таблиць, 5 додатків та 33 використані джерела.

Метою роботи є підвищення ефективності та швидкодії системи комп’ютерного зору шляхом розробки програмного модуля виявлення та відстеження рухомих об’єктів у відеопотоці з використанням можливостей вбудованої обчислювальної платформи BeagleBone AI-64.

Методи дослідження: використано методи системного аналізу та об’єктно-орієнтованого проєктування, алгоритми комп’ютерного зору та машинного навчання, модель YOLOv8 для виявлення об’єктів, алгоритм DeepSORT для відстеження, фільтр Калмана та методи оцінювання продуктивності.

Результати дослідження: розроблено програмний модуль виявлення та відстеження рухомих об’єктів у відеопотоці на платформі BeagleBone AI-64, який поєднує модель YOLOv8 для детекції об’єктів та алгоритм DeepSORT для багатооб’єктного відстеження. Виконано оптимізацію моделі шляхом квантування до формату INT8 та використання апаратних прискорювачів платформи. Проведене експериментальне дослідження підтвердило можливість обробки відеопотоку в режимі реального часу із забезпеченням необхідної точності та продуктивності.

Розроблений програмний продукт може бути використаний у системах відеоспостереження, моніторингу транспортних потоків, робототехнічних комплексах та інших системах комп’ютерного зору, що функціонують на периферійних обчислювальних пристроях.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, ВИЯВЛЕННЯ ОБ’ЄКТІВ, ВІДСТЕЖЕННЯ ОБ’ЄКТІВ, YOLOV8, DEEPSORT, BEAGLEBONE AI-64, ВБУДОВАНІ СИСТЕМИ, МАШИННЕ НАВЧАННЯ.

ANNOTATION

The qualification work entitled “Software Module for Detection and Tracking of Moving Objects in Video Streams on the Embedded BeagleBone AI Platform” for obtaining the Bachelor’s degree in Specialty 122 Computer Science under the educational program Computer Science comprises 74 pages and includes 18 figures, 8 tables, 5 appendices and 33 references.

The aim of the work is to improve the efficiency and performance of a computer vision system by developing a software module for detecting and tracking moving objects in video streams using the capabilities of the embedded BeagleBone AI-64 computing platform.

Research methods: system analysis and object-oriented design methods, computer vision and machine learning algorithms, the YOLOv8 model for object detection, the DeepSORT algorithm for object tracking, the Kalman filter, and performance evaluation methods were used.

Research results: a software module for detecting and tracking moving objects in video streams on the BeagleBone AI-64 platform has been developed. The module combines the YOLOv8 model for object detection and the DeepSORT algorithm for multi-object tracking. The model was optimized through INT8 quantization and the use of hardware accelerators available on the platform. Experimental evaluation confirmed the capability of real-time video stream processing while maintaining the required accuracy and performance.

The developed software solution can be used in video surveillance systems, traffic monitoring applications, robotic systems, and other computer vision solutions operating on edge computing devices.

Keywords: COMPUTER VISION, OBJECT DETECTION, OBJECT TRACKING, YOLOV8, DEEPSORT, BEAGLEBONE AI-64, EMBEDDED SYSTEMS, MACHINE LEARNING..

ЗМІСТ

Вступ 7
1 Теоретичні основи виявлення та відстеження об’єктів 10
1.1 Огляд методів виявлення рухомих об’єктів у відеопотоці 10
1.2 Аналіз алгоритмів відстеження об’єктів у реальному часі 12
1.3 Постановка задачі розробки програмного модуля 19
2 Проєктування програмного модуля виявлення та відстеження 21
2.1 Архітектура програмного модуля та потоки даних 21
2.2 Алгоритмічне забезпечення конвеєра обробки відеопотоку 25
2.3 Вибір та обґрунтування інструментальних засобів розробки 34
2.4 Організація взаємодії програмного модуля з апаратними прискорювачами 37
3 Реалізація та тестування програмного модуля на Beaglebone AI 41
3.1 Програмна реалізація підсистеми виявлення об’єктів 41
3.2 Графічний інтерфейс користувача 49
3.3 Оцінка ефективності та продуктивності програмного модуля 53
Висновки 58
Список використаних джерел 60
Додаток А Копія публікації 64
Додаток Б Доперенавчання моделі YOLOv8 на наборі даних KITTI 69
Додаток В Компіляція та квантування моделі ONNX засобами TIDL 70
Додаток Г Програмна реалізація конвеєра захоплення, обробки та трансляції відеопотоку 72
Додаток Д Програмна реалізація клієнтської підсистеми прийому та декодування відеопотоку 74

ВСТУП

Сучасний етап розвитку інформаційних технологій та інтелектуальних систем характеризується стрімким переходом від централізованих хмарних обчислень до парадигми периферійних обчислень (Edge Computing). Системи відеоспостереження, автономної навігації, розумного міста (Smart City) та робототехніки вимагають обробки масивів візуальних даних безпосередньо на місці їхнього збору. Ключову роль у цих процесах відіграють технології комп'ютерного зору на базі глибоких згорткових нейронних мереж, що забезпечують автоматизоване виявлення та відстеження об'єктів у складних умовах реального середовища.

Незважаючи на значний прогрес у розробці високоточних архітектур машинного навчання, проблема забезпечення їхньої роботи в режимі реального часу на вбудованих платформах залишається відкритою. Перенесення сучасних нейромережевих детекторів та алгоритмів багатометльового відстеження (Multiple Object Tracking) на мікрокомп'ютери супроводжується жорсткими апаратними обмеженнями: лімітованим обсягом оперативної пам'яті, невисокою тактовою частотою ядер центрального процесора та строгими вимогами до енергоспоживання і тепловідведення.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю підвищення ефективності алгоритмів комп'ютерного зору для їхнього функціонування на периферійних пристроях. Вирішення цієї проблеми вимагає розробки спеціалізованих гетерогенних програмних конвеєрів, які здатні ефективно розподіляти обчислювальне навантаження між ядрами загального призначення та матричними прискорювачами (MMA/DSP), а також застосовувати методи оптимізації та цілочисельного квантування нейромережевих моделей без критичної втрати їхньої алгоритмічної точності.

Метою кваліфікаційної роботи є підвищення ефективності та швидкодії системи комп'ютерного зору шляхом розробки та реалізації гібридного програмного модуля виявлення і відстеження рухомих об'єктів, що базується на раціональному розподілі обчислювального навантаження між гетерогенними

ядрами платформи BeagleBone AI-64.

Для досягнення поставленої мети в роботі необхідно розв'язати такі основні завдання:

- провести аналіз сучасних методів виявлення та алгоритмів відстеження об'єктів, а також дослідити архітектурні особливості системи на кристалі TDA4VM платформи BeagleBone AI-64;
- розробити архітектуру багатопроцесорного конвеєра обробки відеопотоку, який поєднує легку згорткову нейронну мережу (YOLOv8) для детекції та оптимізований алгоритм DeepSORT для відстеження;
- здійснити доперенавчання (Transfer Learning) моделі виявлення та виконати її пост-тренувальне квантування (PTQ) для формату INT8 з використанням інструментарію TIDL;
- виконати програмну реалізацію математичного апарату відстеження (фільтр Калмана, обчислення метрик IoU та косинусної відстані) з орієнтацією на мінімізацію накладних витрат центрального процесора;
- провести експериментальне тестування розробленого модуля, оцінити його алгоритмічну точність, пропускну здатність (FPS) та апаратну продуктивність на цільовій платформі.

Об'єктом дослідження є процес виявлення та відстеження рухомих об'єктів у відеопотоці в режимі реального часу.

Предметом дослідження є алгоритми, методи та програмні засоби апаратного прискорення процесу виявлення та відстеження об'єктів, адаптовані для використання на гетерогенній обчислювальній платформі BeagleBone AI-64.

Особливість отриманих результатів полягає в удосконаленні архітектури програмного конвеєра для периферійних систем комп'ютерного зору шляхом комплексної інтеграції апаратного квантування моделей (за допомогою матричних прискорювачів C7x/MMA) з парадигмою багатометльового відстеження через виявлення (DeepSORT). Такий підхід забезпечує стабільну частоту кадрів та мінімізацію наскрізної затримки в умовах строгих апаратних обмежень.

Практична значимість роботи полягає у створенні готового до експлуатації програмного модуля, який ефективно утилізує апаратні можливості платформи

BeagleBone AI-64. Результати дослідження можуть бути безпосередньо використані під час проєктування вузлів автономного вуличного відеоспостереження, систем моніторингу та аналітики транспортного трафіку, а також у комплексах інтелектуальної мобільної робототехніки.

Результати кваліфікаційної роботи були апробовані на IV Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі», яка відбулася 20 травня 2026р. (м. Тернопіль, Україна) та опубліковані у збірнику матеріалів конференції в доповіді «Виявлення та відстеження рухомих об'єктів у відеопотоці на вбудованій обчислювальній платформі Beaglebone AI» . Текст публікації наведено у додатку А.

1 ТЕОРЕТИЧНІ ОСНОВИ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ ОБ’ЄКТІВ

1.1 Огляд методів виявлення рухомих об’єктів у відеопотоці

Виявлення рухомих об’єктів у відеопотоці є однією з найважливіших та базових задач у системах комп’ютерного зору. Цей процес полягає у сегментації пікселів зображення на дві категорії: об'єкти інтересу (передній план) та фон. Від точності та швидкодії алгоритмів на цьому етапі безпосередньо залежить ефективність подальшого відстеження та розпізнавання об’єктів [1].

З огляду на розвиток інформаційних технологій, існуючі методи виявлення можна умовно поділити на дві великі групи: класичні методи обробки зображень та сучасні методи на основі глибокого машинного навчання (рисунок 1.1).

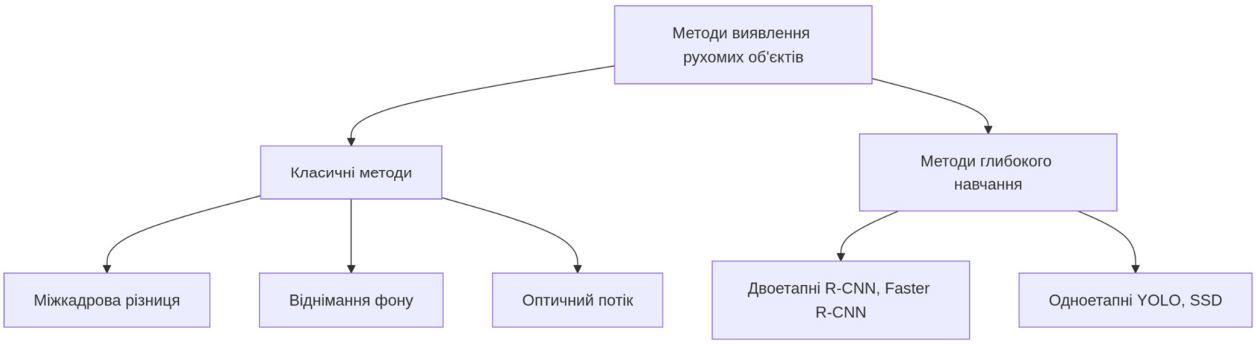

Рисунок 1.1 – Класифікація основних методів виявлення рухомих об’єктів у відеопотоці

Метод міжкадрової різниці (Frame Differencing) Цей підхід є одним із найпростіших та найшвидших в обчислювальному плані. Його суть полягає у попіксельному відніманні двох або більше послідовних кадрів відеопотоку [2]. Математично процес обчислення різниці між поточним та попереднім кадрами описується рівнянням (1.1):

$$D_t(x, y) = |I_t(x, y) - I_{t-1}(x, y)| \quad (1.1)$$

де $D_t(x,y)$ — значення пікселя міжкадрової різниці у момент часу t;

$I_t(x,y)$ — інтенсивність пікселя поточного кадру;

$I_{t-1}(x,y)$ — інтенсивність пікселя попереднього кадру.

Якщо значення $D_t(x,y)$ перевищує заздалегідь задане порогове значення, піксель відноситься до рухомого об'єкта. Попри високу швидкодію, цей метод є вразливим до шумів, зміни освітлення та не здатний ефективно виявляти об'єкти, що тимчасово зупинилися.

Метод віднімання фону (Background Subtraction) є більш стійким підходом та полягає в побудові математичної моделі фону з подальшим порівнянням кожного нового кадру з цією моделлю. Найбільш поширеним алгоритмом у цій категорії є використання суміші гаусіанів (Gaussian Mixture Models, GMM) [3]. На відміну від простої міжкадрової різниці, модель фону постійно оновлюється, що дозволяє адаптуватися до повільних змін освітлення та появи нових статичних об'єктів у кадрі.

Метод оптичного потоку (Optical Flow) Оптичний потік описує видимий рух об'єктів, поверхонь або країв на сцені, викликаний відносним рухом між спостерігачем (камерою) і сценою. Базове рівняння оптичного потоку базується на припущенні про сталість яскравості об'єкта під час його переміщення (1.2):

$$I_x u + I_y v + I_t = 0 \tag{1.2}$$

де I_x, I_y — градієнти яскравості зображення по просторових координатах x та y;

u, v — компоненти вектора швидкості оптичного потоку (шукані величини);

I_t — градієнт яскравості по часу.

Оптичний потік дозволяє отримати не лише інформацію про наявність руху, але й вектор напрямку та швидкості для кожного пікселя [4]. Однак, його обчислення вимагає значних ресурсів, що ускладнює його використання на вбудованих платформах без додаткового апаратного прискорення.

Порівняльна характеристика розглянутих класичних методів наведена у таблиці 1.1.

Таким чином, для вбудованої платформи BeagleBone AI-64 використання виключно класичних методів є компромісом між швидкістю та точністю.

Вирішенням цієї проблеми є перехід до методів на основі згорткових нейронних мереж, які здатні забезпечити високу точність виявлення, а за умови використання вбудованих співпроцесорів — і роботу в режимі реального часу.

Таблиця 1.1 – Порівняльна характеристика класичних методів виявлення рухомих об'єктів

Назва методу	Швидкодія	Стійкість до шумів	Адаптація до зміни освітлення	Обчислювальна складність
Міжкадрова різниця	Висока	Низька	Низька	Низька
Віднімання фону (GMM)	Середня	Середня	Висока	Середня
Оптичний потік	Низька	Висока	Середня	Висока

1.2 Аналіз алгоритмів відстеження об’єктів у реальному часі

Відстеження об’єктів (Object Tracking) є логічним продовженням етапу їхнього виявлення. Основним завданням цього процесу є присвоєння унікального ідентифікатора (ID) кожному виявленому рухомому об'єкту та збереження цього ідентифікатора під час переміщення об'єкта впродовж відеопотоку [5]. Залежно від кількості цілей, задачі відстеження поділяються на відстеження одного об'єкта (Single Object Tracking, SOT) та багатооб'єктне відстеження (Multiple Object Tracking, MOT) [8]. У контексті розробки модуля для платформи BeagleBone AI розглядається саме задача MOT, оскільки в реальних умовах у кадрі зазвичай присутні декілька рухомих об'єктів одночасно.

У сучасних системах комп'ютерного зору найпоширенішою є парадигма «відстеження через виявлення» (Tracking-by-Detection). Вона передбачає незалежне виявлення об'єктів на кожному кадрі з подальшим зв'язуванням цих виявлень у неперервні траєкторії [9].

Одним із базових алгоритмів, що забезпечує високу швидкодію у реальному часі, є алгоритм SORT (Simple Online and Realtime Tracking), який представлений на рисунку 1.2. Цей метод використовує рекурсивний фільтр Калмана (Kalman Filter)

для прогнозування майбутнього положення об'єкта та Угорський алгоритм (Hungarian Algorithm) для оптимального зіставлення спрогнозованих координат із фактичними даними детектора [6].

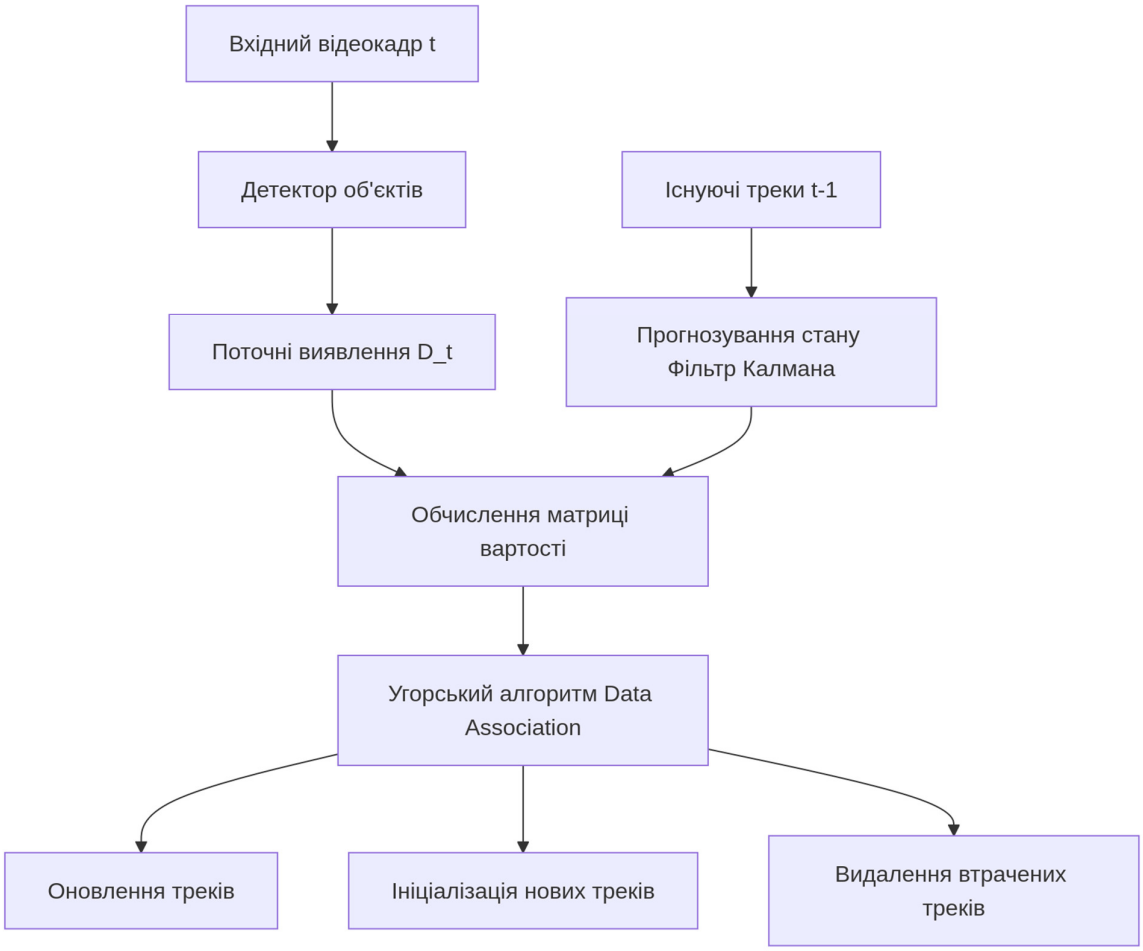

Рисунок 1.2 – Структурна схема парадигми багатооб'єктного відстеження (SORT)

Процес роботи дискретного фільтра Калмана складається з двох основних етапів: прогнозування (Prediction) та корекції (Update). Математична модель етапу прогнозування вектора стану об'єкта описується рівнянням (1.3):

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{1.3}$$

де $\hat{x}_k^-$ - апріорна оцінка вектора стану об'єкта у поточний момент часу *k*;

A — матриця переходу стану системи, що описує кінематику руху;

$\hat{x}_{k-1}$ — апостеріорна оцінка стану у попередній момент часу *k−1*;

B — матриця керування;

u_k — вектор керуючого впливу.

Після отримання нових вимірювань від детектора об'єктів відбувається етап корекції (оновлення) стану за рівнянням (1.4):

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{1.4}$$

де $\hat{x}_k^-$ — апостеріорна (скоригована) оцінка стану;

K_k — коефіцієнт підсилення Калмана (Kalman Gain), що визначає ступінь довіри до нових вимірювань;

z_k — вектор фактичного вимірювання (координати Bounding Box від детектора);

H — матриця спостереження.

Для формування матриці вартості, яку розв'язує Угорський алгоритм, у класичному SORT застосовується метрика перетину площ — Intersection over Union (IoU) . Ця метрика обчислюється як відношення площі перетину двох обмежувальних рамок до площі їх об'єднання:

$$IoU = \frac{Area(B_p \cap B_d)}{Area(B_p \cup B_d)} \tag{1.5}$$

де B_p — спрогнозована обмежувальна рамка (Bounding Box) існуючого треку;

B_d — обмежувальна рамка нового виявлення;

Area — функція обчислення площі прямокутника.

Після зіставлення можливі три результати: 1) оновлення треків (якщо виявлення успішно співставлене з існуючим треком); 2) ініціалізація нових треків (якщо виявлення не було співставлене з жодним треком); 3) видалення втрачених треків (якщо трек не отримував підтвердження протягом певної кількості кадрів).

Основним недоліком алгоритму SORT є втрата ідентифікатора (ID Switch) при частковому або повному перекритті об'єктів (оклюзії) [10]. Для вирішення цієї

проблеми було розроблено алгоритм DeepSORT. Його головна відмінність полягає в інтеграції додаткової метрики зовнішнього вигляду об'єкта (Appearance Descriptor), що отримується за допомогою попередньо навченої глибокої згорткової нейронної мережі. Крім того, просторова відстань між центрами об'єктів оцінюється не лише за IoU, але й з використанням відстані Махаланобіса, що враховує невизначеність оцінки стану фільтром Калмана [7].

Для оцінки візуальної схожості об'єктів у DeepSORT використовується косинусна відстань, що розраховується за рівнянням:

$$d^{(2)}(i,j) = 1 - r_j^T f_i \tag{1.6}$$

де $d^{(2)}(i,j)$ — косинусна відстань між i-м виявленим об'єктом та j-м існуючим треком;

r_j — вектор візуальних ознак j-го треку, збережений у галереї зразків;

f_i — вектор візуальних ознак поточного i-го виявлення.

Порівняльна характеристика розглянутих алгоритмів відстеження наведена у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика алгоритмів SORT та DeepSORT

Критерій порівняння	Алгоритм SORT	Алгоритм DeepSORT
Використання візуальних ознак	Не використовується	Використовується (CNN дескриптор)
Метрика зіставлення	Тільки IoU	IoU, відстань Махаланобіса та косинусна відстань
Стійкість до перекриття (оклюзії)	Низька	Висока
Кількість втрат ID (ID Switches)	Висока	Значно знижена
Вимоги до обчислювальних ресурсів	Низькі	Середні/Високі (залежить від моделі CNN)
Доцільність для BeagleBone AI	Висока (максимальна швидкодія)	Середня (потребує апаратного прискорення моделі)

Таким чином, аналіз показує, що вибір між класичним SORT та DeepSORT для вбудованої обчислювальної платформи BeagleBone AI є складним компромісом між ресурсною ефективністю та стійкістю процесу відстеження. Найбільш раціональним рішенням для поставленої задачі є розробка гібридного підходу на базі логіки DeepSORT із застосуванням оптимізованої полегшеної моделі виділення ознак. Це дозволить ефективно задіяти спеціалізовані апаратні прискорювачі платформи без суттєвої деградації частоти кадрів (FPS) відеопотоку.

Успішна реалізація методів виявлення та відстеження об'єктів у реальному часі вимагає значних обчислювальних ресурсів. Для вбудованих систем базових потужностей центрального процесора (CPU) зазвичай недостатньо для забезпечення високої частоти кадрів (FPS) під час виконання інференсу глибоких нейронних мереж. Платформа BeagleBone AI-64 базується на гетерогенній системі на кристалі (SoC) Texas Instruments TDA4VM, мікроархітектура якої спеціально спроєктована для задач комп'ютерного зору та аналітики на периферійних пристроях (Edge AI) [11]. До складу обчислювальної системи TDA4VM, окрім двох потужних універсальних ядер ARM Cortex-A72 (64-bit), входять два цифрові сигнальні процесори C66x та спеціалізований процесор цифрової обробки сигналів нового покоління C7x, який містить інтегрований матричний прискорювач MMA (Matrix Multiply Accelerator). Наявність блоку MMA дозволяє виконувати ресурсомісткі тензорні операції з продуктивністю до 8 TOPS (Trillion Operations Per Second), суттєво розвантажуючи центральний процесор для задач керування потоками даних, роботи з операційною системою Linux та алгоритмів високого рівня (наприклад, фільтра Калмана в алгоритмі SORT) [12]. Архітектура ядра C7x базується на 64-розрядному VLIW (Very Long Instruction Word) процесорі, який поєднує можливості традиційного DSP із суперскалярною обробкою. Тісно інтегрований з ним співпроцесор MMA спеціалізується на надшвидкому виконанні щільних матричних множень із накопиченням (Multiply-Accumulate, MAC), що лежать в основі згорткових шарів (Convolutional Layers) нейронних мереж [13]. Для уникнення проблеми «вузького місця» пропускної здатності пам'яті (Memory Wall) під час постійного завантаження векторних конвеєрів, платформою передбачено використання вдосконаленого контролера прямого доступу до пам'яті (Enhanced

Direct Memory Access, EDMA) та багаторівневої системи кешування. Систему передачі даних між компонентами платформи відображено на рисунку 1.3.

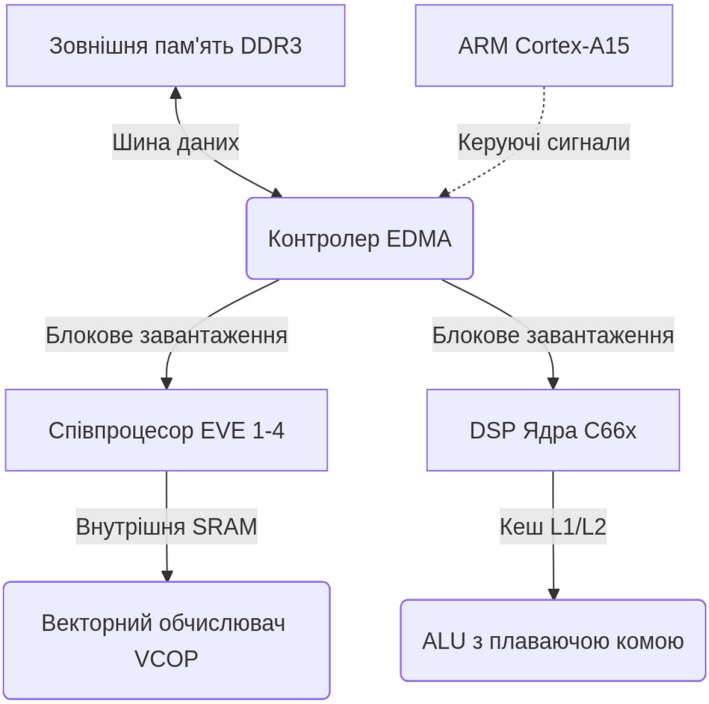

Рисунок 1.3 – Схема організації потоків даних та доступу до пам'яті на SoC TDA4VM

Для ефективного використання апаратного прискорювача C7x/MMA застосовується спеціалізований програмний стек TIDL (Texas Instruments Deep Learning). Його першочерговим завданням є трансляція та оптимізація обчислювального графа попередньо навченої моделі (наприклад, з форматів TensorFlow або ONNX). Ключовим етапом підготовки моделі у TIDL є процес квантування (Quantization). Оскільки прискорювач MMA досягає максимальної продуктивності та енергоефективності під час роботи з цілочисельними даними розрядності 8 біт (INT8), вагові коефіцієнти та активації моделі, які початково представлені у форматі з плаваючою комою (FP32), необхідно дискретизувати.

Математична модель афінного (асиметричного) квантування описується рівнянням (1.7):

$$q = \text{round}\left(\frac{r}{S}\right) + Z \tag{1.7}$$

де q — квантоване цілочисельне значення (у форматі INT8);

r — дійсне значення вхідного тензора (у форматі FP32);

S — масштабний коефіцієнт (Scale), що визначає крок квантування;

Z — точка нуля (Zero-point), що компенсує зміщення діапазону дійсних значень.

Для оптимізації швидкодії TIDL також виконує злиття шарів (Layer Fusion). Зокрема, шар пакетної нормалізації математично інтегрується (зливається) з попереднім згортковим шаром Перерахунок вагових коефіцієнтів W_{fused} та зсуву b_{fused} описується системою рівнянь (1.8):

$$W_{fused} = \frac{\gamma W}{\sqrt{\sigma^2 + \epsilon}}, \quad b_{fused} = \frac{\gamma(b - \mu)}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{1.8}$$

де *W, b* — початкові вагові коефіцієнти та вектор зсуву згорткового шару;

μ, σ^2 — середнє значення та дисперсія пакетної нормалізації;

γ, β — параметри масштабування та зсуву;

є — математична константа для уникнення ділення на нуль.

Таке математичне перетворення дозволяє повністю вилучити шар пакетної нормалізації з графа інференсу, значно скоротивши кількість операцій читання/запису пам'яті. Використання 8-бітного квантування зменшує обсяг оперативної пам'яті для зберігання моделі у 4 рази, що є критичним фактором для вбудованих платформ з обмеженими ресурсами.

Оптимальний розподіл обчислювального навантаження розробленого модуля між гетерогенними ядрами платформи наведено у таблиці 1.3.

Підсумовуючи, архітектурні особливості BeagleBone AI-64 вимагають специфічного підходу до розробки програмного забезпечення. Забезпечення роботи модуля у реальному часі можливе лише за умови перенесення важких згорткових обчислень на спеціалізований прискорювач C7x/MMA за допомогою інструментарію TIDL та Edge AI SDK, а також попереднього квантування моделей до цілочисельного формату.

Таблиця 1.3 – Розподіл навантаження на платформі BeagleBone AI-64

Обчислювальний вузол	Підтримувані типи даних	Функціональне призначення у програмному модулі
ARM Cortex-A72 (64-bit)	FP64, FP32, INT8	Виконання ОС Linux, захоплення відеокадрів, логіка трекінгу (алгоритм Угорських призначень)
C66x DSP	FP32, FP16, INT8	Обчислення косинусної метрики схожості, допоміжні математичні операції
C7x DSP + MMA	INT8, INT16, FP32	Виконання щільних згорткових шарів моделі виявлення об'єктів та екстракції ознак із продуктивністю до 8 TOPS

1.3 Постановка задачі розробки програмного модуля

Стрімкий розвиток систем інтелектуального відеоспостереження, робототехніки та автономних апаратів вимагає перенесення обчислювально складних процесів обробки відеоданих із централізованих хмарних серверів безпосередньо на кінцеві периферійні пристрої (Edge Computing) [14]. Такий підхід дозволяє мінімізувати затримки передачі даних, значно знизити навантаження на телекомунікаційні канали зв'язку та забезпечити повну автономність роботи системи у випадку відсутності або нестабільності підключення до мережі Інтернет. Однак практична реалізація сучасних алгоритмів комп'ютерного зору на вбудованих обчислювальних платформах, таких як BeagleBone AI-64, стикається з низкою жорстких апаратних обмежень: лімітованим обсягом оперативної пам'яті, порівняно невисокою тактовою частотою ядер центрального процесора та строгими вимогами до енергоспоживання і тепловідведення.

На основі детального аналізу, проведеного у попередніх підрозділах, можна констатувати наявність суттєвої науково-практичної суперечності. З одного боку, класичні методи комп'ютерного зору (віднімання фону, оптичний потік) здатні працювати у реальному часі на слабкому апаратному забезпеченні, проте вони не забезпечують необхідного рівня точності виявлення об'єктів в умовах динамічного освітлення, наявності шумів та перекриттів (оклюзій). З іншого боку, використання виключно глибоких згорткових нейронних мереж (CNN) як для задачі детекції, так

і для задачі безперервного відстеження дозволяє досягти високої точності, але призводить до критичного падіння частоти кадрів (Frames Per Second, FPS), що унеможливлює використання системи у режимі реального часу без застосування методів апаратного прискорення та алгоритмічної оптимізації.

З огляду на зазначене, об'єктом дослідження є процес виявлення та відстеження рухомих об'єктів у відеопотоці в режимі реального часу.

Предметом дослідження є алгоритми, методи та програмні засоби апаратного прискорення процесу виявлення та відстеження рухомих об'єктів, адаптовані для використання на вбудованій обчислювальній платформі BeagleBone AI-64.

Метою кваліфікаційної роботи є підвищення ефективності та швидкодії системи комп'ютерного зору шляхом розробки та реалізації гібридного програмного модуля виявлення і відстеження об'єктів, що базується на раціональному розподілі обчислювального навантаження між гетерогенними ядрами платформи (ARM Cortex-A72, C7x DSP, C66x DSP) та квантуванні моделей нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такий комплекс взаємопов'язаних завдань:

1. Здійснити аналіз сучасних методів виявлення та алгоритмів відстеження об'єктів (зокрема SORT та DeepSORT), а також дослідити архітектурні особливості системи на кристалі TDA4VM платформи BeagleBone AI-64.
2. Розробити архітектуру гібридного конвеєра обробки відеопотоку, який поєднає легку згорткову нейронну мережу для детекції (YOLO або SSD) та оптимізований алгоритм DeepSORT для трекінгу.
3. Провести адаптацію та квантування вибраних моделей глибокого навчання (зниження розрядності вагових коефіцієнтів з FP32 до INT8) з використанням інструментарію оптимізації Texas Instruments Deep Learning (TIDL) для їхнього виконання на матричному прискорювачі MMA та ядрі C7x.
4. Здійснити програмну реалізацію математичного апарату відстеження (фільтр Калмана, обчислення метрик IoU та косинусної відстані, алгоритм лінійного призначення) з орієнтацією на виконання на ядрах ARM та DSP.
5. Провести експериментальне тестування розробленого модуля, оцінити його алгоритмічну точність та апаратну продуктивність на цільовій платформі.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ

2.1 Архітектура програмного модуля та потоки даних

Для забезпечення ефективної роботи програмного модуля виявлення та відстеження об'єктів у режимі реального часу на вбудованій платформі BeagleBone AI-64, розробка має базуватися на ретельно спроєктованій програмній архітектурі. Зважаючи на гетерогенність обчислювальної системи (наявність ядер ARM Cortex-A72, цифрових сигнальних процесорів DSP та матричного прискорювача MMA), класична послідовна (синхронна) архітектура виконання є вкрай неефективною, оскільки вона призводить до простою одних обчислювальних вузлів під час роботи інших [17].

В основу архітектури розроблюваного модуля покладено модульний підхід та конвеєрний шаблон проєктування (Pipeline Design Pattern) з використанням багатопотоковості. Система розділена на логічно ізольовані компоненти (модулі), взаємодія між якими відбувається через потокобезпечні черги даних (Thread-safe Queues) за принципом «виробник-споживач» (Producer-Consumer). Це дозволяє розпаралелити процеси захоплення кадрів, їх обробки нейромережею та логіки відстеження [18].

Через комплексну природу програмного конвеєра, загальну архітектуру модуля доцільно декомпозувати на два взаємопов'язані етапи. Перший етап (рисунок 2.1) відповідає за введення відеоданих, їхню підготовку та апаратну детекцію регіонів інтересу.

До складу першого етапу входять дві базові підсистеми, які функціонують асинхронно:

- підсистема захоплення відео (Video Capture Subsystem): Відповідає за ініціалізацію відеоджерела (USB-камера, CSI-камера або відеофайл), декодування потоку та формування оригінального кадру. Працює в окремому потоці (Thread 1) для мінімізації затримок введення-виведення (I/O). Сформована матриця пікселів передається до проміжної потокобезпечної черги кадрів.
- підсистема детекції об'єктів (Detection Subsystem): Зчитує оригінальні

кадри з черги, виконує їхню попередню обробку (зміну розміру, нормалізацію колірного простору з BGR у RGB) та формує вхідний тензор. Далі модуль викликає API інференсу (TIDL / TI Edge AI), делегуючи виконання важких операцій згорткової нейромережі потужному спеціалізованому прискорювачу C7x/MMA. На виході підсистема (Thread 2) генерує вектор виявлених об'єктів (Bounding Boxes) з їхніми координатами та класами. Сформований на першому етапі вектор детекцій передається до другого етапу конвеєра (рисунок 2.2), який відповідає за просторово-часову асоціацію виявлених об'єктів та фінальне формування графічного відображення. Сформований на першому етапі вектор детекцій передається до другого етапу конвеєра (рисунок 2.2), який відповідає за просторово-часову асоціацію виявлених об'єктів та фінальне формування графічного відображення.

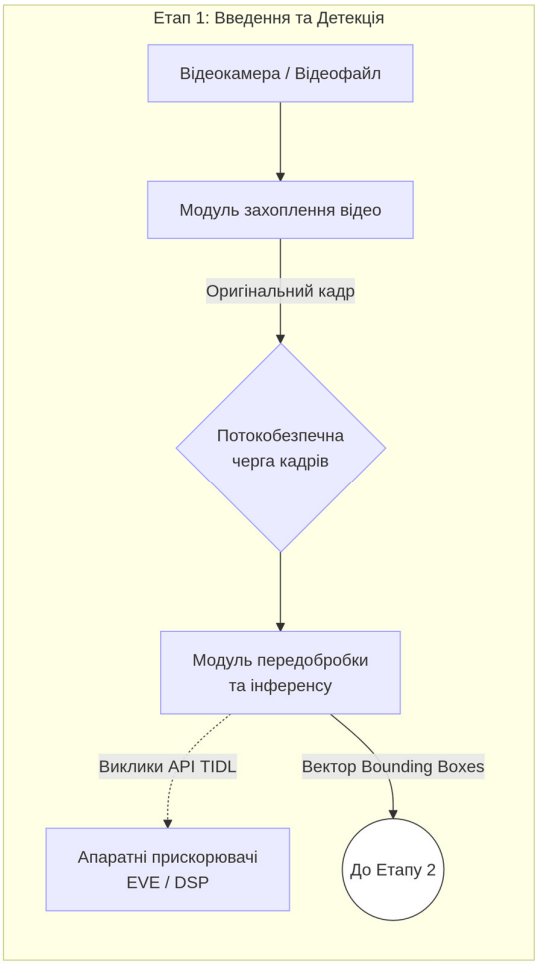

Рисунок 2.1 – Етап 1: Конвеєр захоплення відеоданих та детекції об'єктів

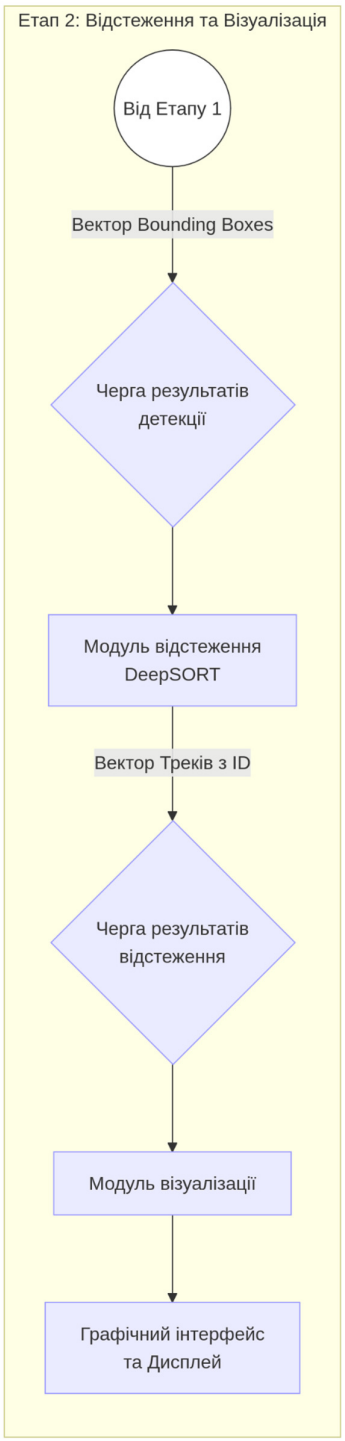

Рисунок 2.2 – Етап 2: Конвеєр відстеження об'єктів (DeepSORT) та візуалізації

До складу другого етапу входять наступні підсистеми:

- підсистема відстеження (Tracking Subsystem): Приймає координати з черги результатів детекції, розраховує візуальні дескриптори за допомогою легкої CNN для екстракції ознак, оновлює стан фільтрів Калмана для існуючих треків та виконує асоціацію даних. Робота Угорського алгоритму та логіки DeepSORT

виконується переважно на ядрах загального призначення ARM Cortex-A72 (Thread 3). Результатом є вектор активних треків із присвоєними унікальними ідентифікаторами (ID).

- підсистема постобробки та візуалізації (Visualization Subsystem): Отримує дані з черги результатів відстеження, накладає відповідні графічні примітиви (рамки, ідентифікатори, траєкторії руху) на оригінальний кадр та виводить готовий результат на графічний інтерфейс дисплея або транслює у мережу (Thread 4). Оскільки архітектура є асинхронною, загальна продуктивність системи (пропускна здатність) визначається не сумою часу виконання всіх модулів, а часом виконання найповільнішого етапу конвеєра (вузького місця). Математично максимальна частота кадрів відеопотоку FPS_max, яку здатна обробити система, обчислюється за формулою (2.1):

$$FPS_{max} = \frac{1}{\max(T_{cap}, T_{det}, T_{trk}, T_{vis})} \tag{2.1}$$

де T_{cap} — середній час захоплення та декодування одного відеокадру;

T_{det} — час роботи підсистеми детекції (включаючи час інференсу на матричному прискорювачі MMA);

T_{trk} — час виконання алгоритмів відстеження об'єктів;

T_{vis} — час рендерингу графіки та виведення кадру на дисплей.

З метою синхронізації роботи потоків та уникнення стану гонитви (Race Condition) при зверненні до спільної пам'яті, застосовується механізм м'ютексів (Mutex) та умовних змінних (Condition Variables). Якщо черга переповнюється (наприклад, алгоритм детекції не встигає обробляти кадри від камери), найстаріші кадри відкидаються (Frame Dropping) для забезпечення актуальності даних у реальному часі. Характеристика інтерфейсів передачі даних між основними підсистемами наведена у таблиці 2.1.

Проєктування інформаційного забезпечення також включає роботу з конфігураційними файлами (формату JSON або YAML), які дозволяють гнучко налаштовувати гіперпараметри системи без необхідності перекомпіляції коду. До

таких параметрів належать: шляхи до вагових коефіцієнтів моделей (TIDL artifacts), пороги впевненості детектора (Confidence Threshold), параметри матриць коваріації фільтра Калмана та максимальна кількість кадрів, протягом яких система зберігає ідентифікатор об'єкта у стані невидимості (Max Age).

Таблиця 2.1 – Інтерфейси взаємодії модулів у програмному конвеєрі

Відправник (Producer)	Отримувач (Consumer)	Структура даних (Payload)	Опис переданих даних
Підсистема захоплення	Підсистема детекції	cv::Mat (або numpy.ndarray), uint64_t	Багатовимірний масив пікселів кадру, унікальний номер кадру (Timestamp)
Підсистема детекції	Підсистема відстеження	std::vector<Detection>, cv::Mat	Масив структур (координати x,y,w,h, рівень впевненості conf, клас class_id) та оригінальний кадр для екстракції ознак
Підсистема відстеження	Підсистема візуалізації	std::vector<Track>	Масив активних треків (унікальний track_id, згладжені координати, історія переміщень)

Отже, запропонована архітектура програмного модуля забезпечує високий ступінь модульності, дозволяє максимально завантажити гетерогенні обчислювальні ядра BeagleBone AI-64 та гарантує стабільну обробку відеопотоку завдяки використанню асинхронних черг повідомлень.

2.2 Алгоритмічне забезпечення конвеєра обробки відеопотоку

Алгоритмічне забезпечення розробленого програмного модуля визначає строгу послідовність математичних та логічних операцій, які ітеративно виконуються над кожним відеокадром від моменту його захоплення до моменту виведення оновлених траєкторій рухомих об'єктів. Зважаючи на використання гібридної парадигми «відстеження через виявлення» (Tracking-by-Detection), загальний цикл обробки для кожного моменту часу t концептуально поділяється на три взаємопов'язані стадії: виявлення просторових координат, екстракція ознак і

прогнозування, а також асоціація даних [20].

Структурна блок-схема розробленого алгоритму обробки одного відеокадру через свою складність декомпозована на дві логічні підсистеми. Перша підсистема відповідає за виявлення об'єктів та підготовку їхніх візуальних ознак (рисунок 2.3).

Рисунок 2.3 – Блок-схема алгоритму виявлення та екстракції ознак об'єктів

Як видно з рисунка 2.3, на початковому етапі алгоритм функціонує виключно у просторовому домені поточного кадру. Головним завданням цієї частини конвеєра є мінімізація хибних спрацьовувань (False Positives) детектора за рахунок жорсткої порогової фільтрації та алгоритму NMS. Сформовані вектори ознак (дескриптори) разом із координатами передаються до другої частини алгоритму.

Першим алгоритмічним етапом є підготовка вхідних даних та безпосереднє виявлення об'єктів. Оригінальний кадр I_t, отриманий від підсистеми захоплення відео, як правило, має високу роздільну здатність (наприклад, 1920×1080 пікселів), що є надмірним для апаратних прискорювачів платформи BeagleBone AI-64. Тому кадр проходить процедуру білінійної інтерполяції (зменшення до розмірів вхідного тензора 300×300 або 320×320 пікселів) та попіксельної нормалізації (віднімання середніх значень колірних каналів для зведення діапазону [0, 255] до [-1, 1]).

Підготовлений тензор передається до квантованої згорткової нейронної мережі (наприклад, MobileNet-SSD), яка формує множину «сирих» прогнозів (Raw Detections). Для усунення надлишкових спрацьовувань застосовується двоступенева фільтрація.

На першому кроці відкидаються всі обмежувальні рамки b_i, рівень впевненості яких $con\ f(b_i)$ нижчий за емпірично встановлений поріг $\tau_{con\ f}$. Множина валідних кандидатів B визначається за рівнянням (2.2):

$$B = \{b_i \mid conf(b_i) \geq \tau_{conf},\ i = 1, \ldots, N\} \tag{2.2}$$

Оскільки навколо одного фізичного об'єкта детектор зазвичай генерує кілька рамок, на другому кроці застосовується алгоритм придушення немаксимумів (Non-Maximum Suppression, NMS). Алгоритм сортує множину B за спаданням рівня впевненості. Рамка M з найвищим показником $con\ f$ фіксується як еталонна, а всі інші рамки b_i, площа перетину яких із рамкою M перевищує поріг τ_{nms}, видаляються [21].

Метрика перетину площ (Intersection over Union, IoU) обчислюється за формулою (2.3):

$$IoU(M, b_i) = \frac{Area(M \cap b_i)}{Area(M \cup b_i)} \geq \tau_{nms} \tag{2.3}$$

У результаті етапу детекції формується очищений масив виявлень $D_t=\{d_1,d_2,\ldots,d_m\}$, де кожне виявлення d_j характеризується координатами центру, шириною та висотою (c_x,c_y,w,h).

Для забезпечення стійкості відстеження в умовах перекриттів (оклюзій), система не обмежується лише просторовими координатами. Кожна обмежувальна рамка з множини D_t вирізається з оригінального кадру та подається на вхід спеціалізованої нейромережі екстракції ознак (Re-ID моделі). На виході формується вектор візуальних ознак (дескриптор) r_j розмірністю 128 або 256 елементів. Модель проектується таким чином, щоб вектори ознак одного і того ж об'єкта були близькими у багатовимірному просторі, а різних об'єктів — далекими (використовується функція втрат Triplet Loss під час навчання) [22].

Паралельно з цим, для кожного існуючого активного треку t_i з попереднього кадру активується етап прогнозування дискретного фільтра Калмана. Стан об'єкта у просторі моделюється як 8-вимірний вектор.

Використовуючи стандартну кінематичну модель рівномірного прямолінійного руху, фільтр Калмана розраховує апріорну оцінку положення об'єкта на поточному кадрі t та коваріаційну матрицю похибки P, яка відображає ступінь невизначеності цього прогнозу.

Логіка другої підсистеми алгоритму — безпосереднього відстеження та асоціації даних у часі — наведена на рисунку 2.4.

Центральним ядром підсистеми відстеження є задача зіставлення множини нових виявлень D_t із множиною спрогнозованих треків T_{t-1}. Цей процес моделюється як класична задача про лінійне призначення (Linear Assignment Problem).

Спершу формується матриця вартості C розмірністю N×M (де N — кількість треків, M — кількість виявлень). У розробленому гібридному алгоритмі кожен елемент матриці $C_{i,j}$ є зваженою сумою двох незалежних метрик (2.4):

$$C_{i,j} = \lambda \cdot D_M(t_i, d_j) + (1 - \lambda) \cdot D_C(t_i, d_j) \tag{2.4}$$

де D_M — квадрат відстані Махаланобіса між прогнозованим станом треку t_i та виміряними координатами виявлення d_j. Вона враховує ймовірність (матрицю P) фільтра Калмана і ефективно відтинає неможливі переміщення;

D_C — косинусна відстань між векторами візуальних ознак (дескрипторами), що дозволяє ідентифікувати об'єкт навіть після тривалого перекриття;

λ — ваговий гіперпараметр (зазвичай λ=0.5).

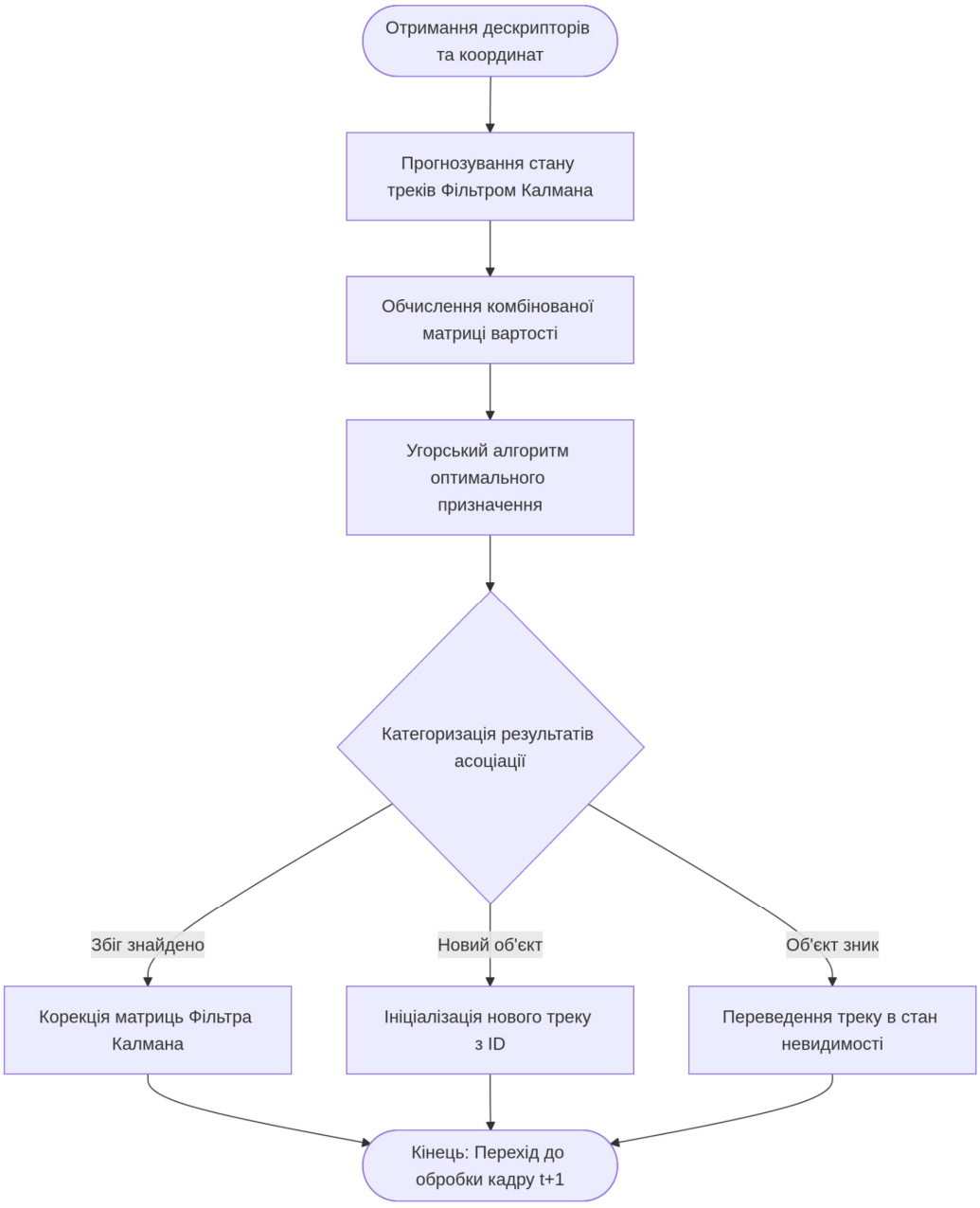

Рисунок 2.4 – Структурна схема взаємодії програмних інструментальних засобів з апаратним забезпеченням

Для мінімізації загальної вартості призначень до матриці C застосовується Угорський алгоритм (алгоритм Куна-Мункреса) [23]. На основі отриманих результатів та подальшої порогової фільтрації (відхилення призначень, вартість яких перевищує допустимий ліміт) утворюються три категорії об'єктів, для кожної з яких застосовується власна логіка оновлення життєвого циклу [24]:

1. Зіставлені пари (Matched Tracks): Виявлення, яким успішно знайдено відповідний трек. Їхні нові координати використовуються для корекції (оновлення) стану фільтра Калмана. Дескриптор об'єкта додається до галереї візуальних ознак цього треку, а лічильник кадрів без видимості (time_since_update) скидається до нуля.

2. Незіставлені виявлення (Unmatched Detections): Вважаються новими об'єктами, щойно з'явилися в кадрі. Для них ініціалізуються нові фільтри Калмана. Треку надається статус «Попередній» (Tentative). Якщо він підтверджується на наступних k кадрах, він переходить у статус «Підтверджений» (Confirmed) і йому присвоюється новий унікальний ID.

3. Незіставлені треки (Unmatched Tracks): Об'єкти, що залишили кадр або були загороджені іншими перешкодами. Алгоритм продовжує прогнозувати їхнє положення «наосліп», щоразу збільшуючи лічильник time_since_update. Якщо цей лічильник перевищує поріг максимального віку (наприклад, MaxAge=30 кадрів), трек остаточно переводиться у статус «Видалений» (Deleted) і вивільняє оперативну пам'ять.

Таке багаторівневе алгоритмічне забезпечення створює стійкий і замкнений цикл обробки, який мінімізує кількість втрат ідентифікаторів (ID Switches) і здатний надійно функціонувати в режимі реального часу.

Для комплексного відображення принципів роботи розробленого програмного модуля побудовано низку системних моделей, які деталізують його функціональну, структурну та інформаційну організацію.

Ієрархію завдань, які ітеративно виконує система для досягнення головної мети — виявлення та відстеження рухомих об'єктів у відеопотоці, — відображає функціональна схема (рисунок 2.5). Вона демонструє логічну декомпозицію процесу на базові функції: початкову ініціалізацію, апаратне захоплення відео, детекцію (з

делегуванням обчислень на спеціалізований C7x DSP із матричним прискорювачем MMA), відстеження (асоціацію даних) та фінальну візуалізацію результатів.

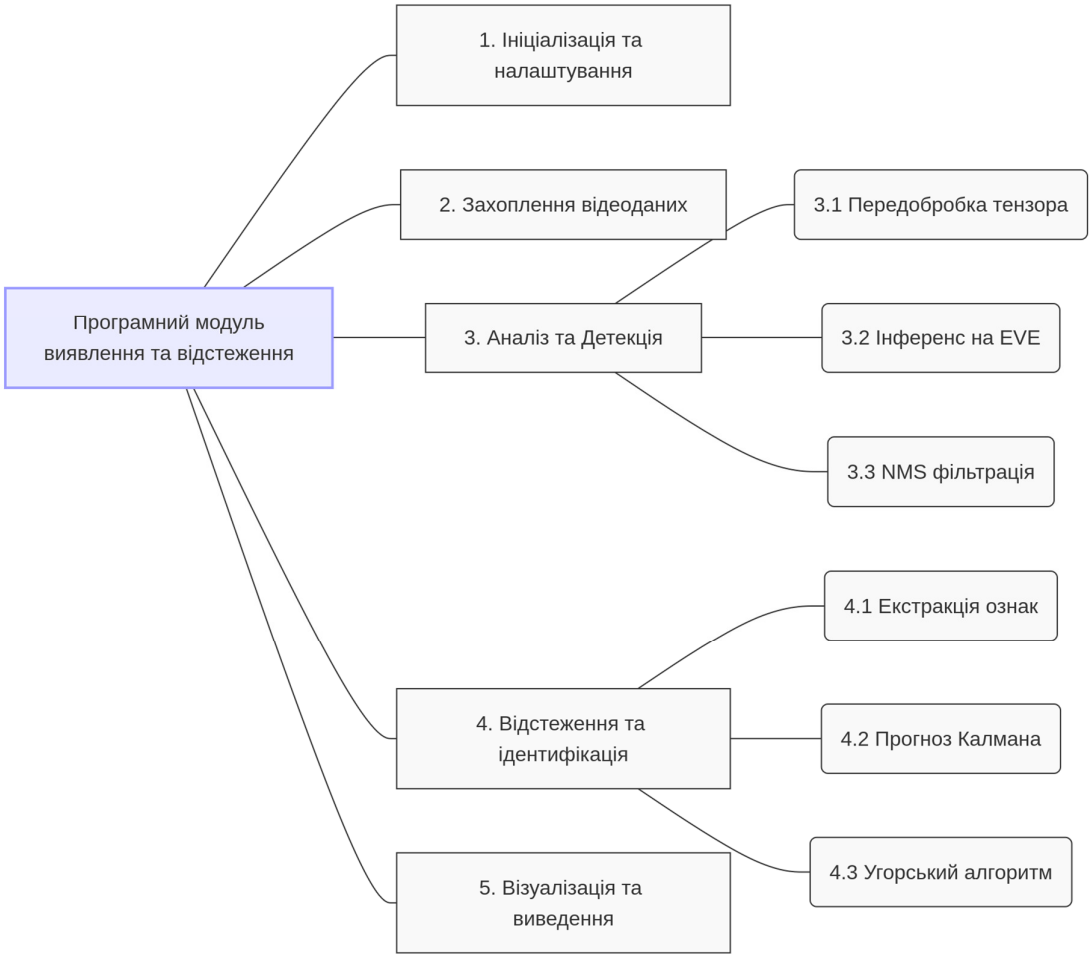

Рисунок 2.5 – Функціональна схема програмного модуля

Логічна та фізична організація компонентів модульної системи наведена на структурній схемі (рисунок 2.6). Запропонована архітектура відображає чітке вертикальне розмежування на прикладний рівень (багатопотоковий додаток мовою C++), системний рівень (ОС Linux, відеодрайвер V4L2, програмний інтерфейс TIDL API, драйвер безперервної пам'яті CMEM) та апаратний рівень (ядра загального призначення ARM Cortex-A72 (64-bit), спеціалізований DSP-процесор архітектури C7x із прискорювачем глибокого навчання MMA (Matrix Multiply Accelerator) та цифрові сигнальні процесори C66x мікропроцесора TDA4VM).

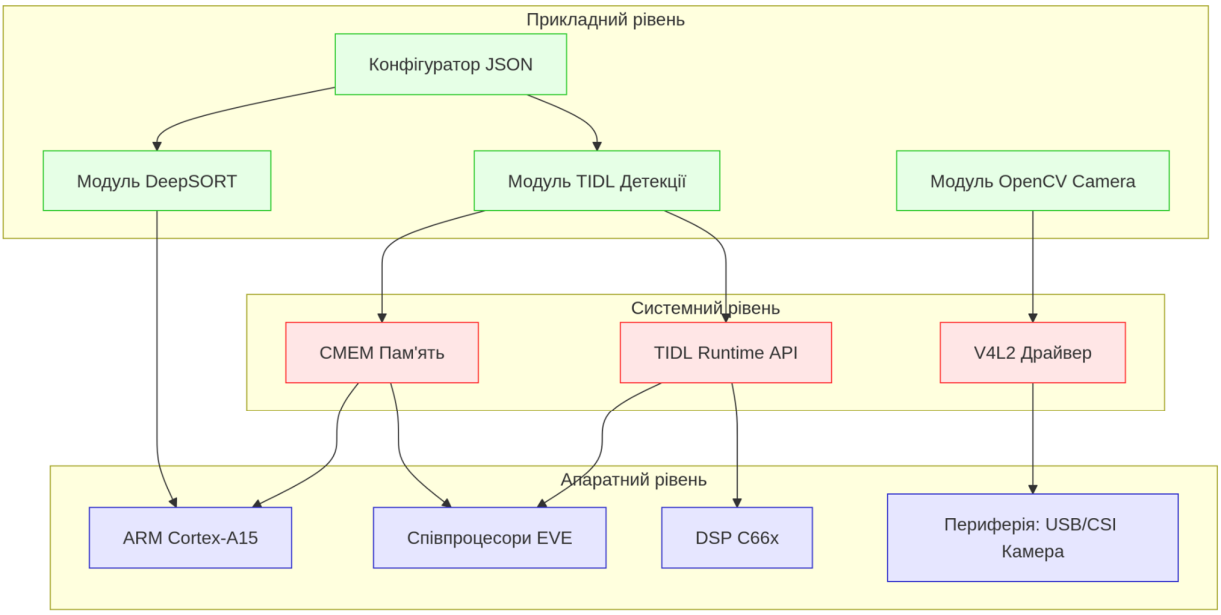

Рисунок 2.6 – Структурна схема системи на базі SoC TDA4VM

Оскільки система проєктується як Edge AI рішення для роботи в автономному режимі на вбудованій платформі, втручання людини у процес обробки є мінімальним. Діаграма прецедентів (Use Case), наведена на рисунку 2.7, ілюструє основні варіанти використання та сценарії взаємодії адміністратора з системою. Вони переважно зводяться до початкового налаштування гіперпараметрів (через JSON-конфігурацію), ініціалізації конвеєра та візуального моніторингу результатів відстеження.

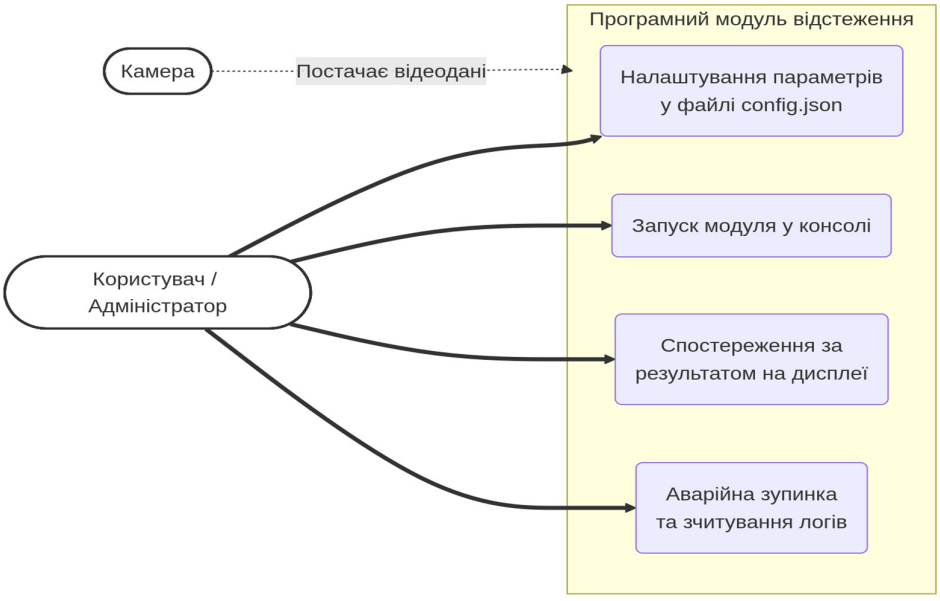

Рисунок 2.7 – Діаграма прецедентів взаємодії користувача з системою

Маршрутизацію інформації між обчислювальними процесами, сховищами (пам'яттю) та зовнішніми пристроями відображає діаграма потоків даних (Data Flow Diagram, DFD) першого рівня (рисунок 2.8). Вона візуалізує трансформацію вхідної інформації під час проходження через конвеєр: від сирого потоку байтів камери до багатовимірних тензорів у пам'яті CMEM, і, зрештою, до вектора ідентифікаторів об'єктів (Track ID), який накладається на вихідний відрендерений кадр.

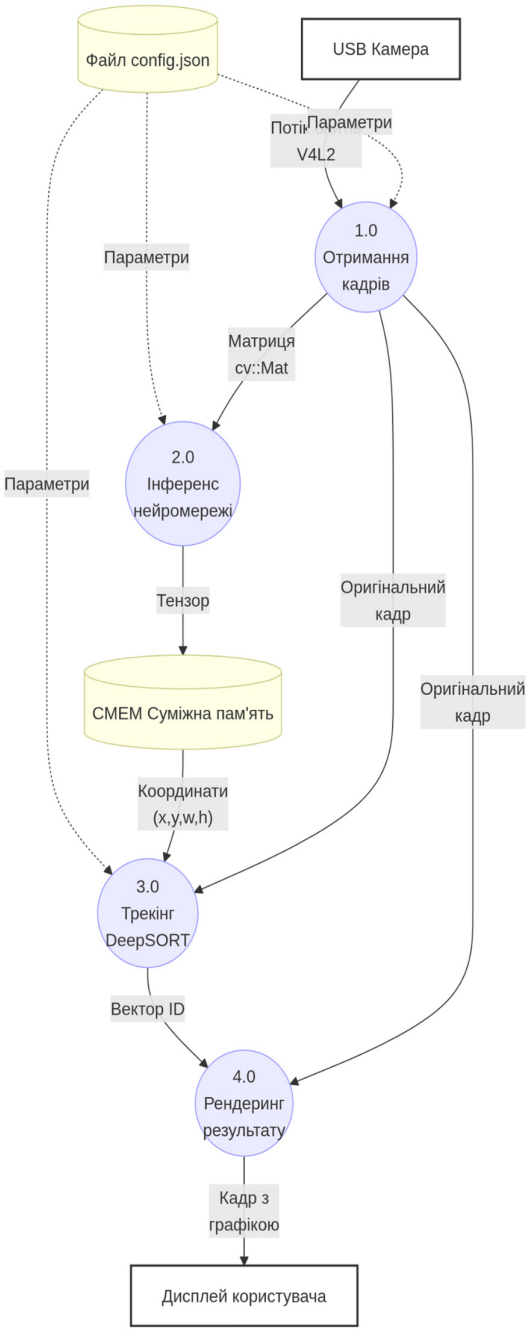

Рисунок 2.8 – Діаграма потоків даних програмного конвеєра

2.3 Вибір та обґрунтування інструментальних засобів розробки

Ефективність функціонування розроблених алгоритмів виявлення та відстеження об'єктів у режимі реального часу залежить не лише від мікроархітектури апаратної платформи, але й від правильного вибору стека програмних технологій. Програмне забезпечення вбудованих систем (Embedded Systems) функціонує в умовах жорстких обмежень щодо обсягу оперативної пам'яті, енергоспоживання та обчислювальної потужності. З огляду на це, вибір інструментальних засобів розробки має базуватися на критеріях максимальної швидкодії, можливості низькорівневого керування пам'яттю та наявності оптимізованих компіляторів для архітектури AArch64 (ARM64) [25].

Для реалізації завдань машинного навчання та комп'ютерного зору на сьогодні найпопулярнішими мовами програмування є Python та C++. Мова Python володіє багатим спектром бібліотек для створення прототипів нейронних мереж, однак її використання як цільової мови для виконання інференсу на пристроях Edge AI (до яких належить BeagleBone AI-64) є недоцільним з кількох причин. По-перше, наявність глобального блокування інтерпретатора (Global Interpreter Lock, GIL) унеможливлює повноцінну реалізацію справжньої багатопотоковості, що критично важливо для спроєктованої у підрозділі 2.1 конвеєрної архітектури. По-друге, автоматичне збирання сміття (Garbage Collection) вносить недетерміновані затримки у процес обробки відеокадрів.

На основі проведеного аналізу, для реалізації програмного модуля нами було обрано мову програмування C++ (стандарту C++14/17). Ця мова забезпечує прямий доступ до апаратних ресурсів пам'яті, дозволяє здійснювати ручне керування життєвим циклом об'єктів та гарантує мінімальні накладні витрати часу (Overhead) під час виконання циклів [26].

Основним інструментом для виконання операцій передобробки та візуалізації відеоданих обрано бібліотеку OpenCV (Open Source Computer Vision Library). Це найбільш функціональний фреймворк комп'ютерного зору, написаний на оптимізованому C/C++. Важливою перевагою OpenCV для платформи BeagleBone AI-64 є підтримка інструкцій векторного співпроцесора NEON (Advanced SIMD),

вбудованого у ядра ARM Cortex-A72 (64-bit).

У розроблюваному модулі планується використання таких базових модулів бібліотеки OpenCV:

- core — для роботи з багатовимірними масивами cv::Mat (внутрішнє представлення відеокадрів та тензорів);
- videoio — для апаратного захоплення відеопотоку з USB-камери через інтерфейс V4L2 (Video for Linux 2);
- imgproc — для білінійної інтерполяції (масштабування) кадрів, конвертації колірних просторів (BGR у RGB) та накладання обмежувальних рамок на фінальне зображення.

Оскільки OpenCV не має вбудованої підтримки специфічних прискорювачів C7x/MMA та DSP платформи TDA4VM, для виконання інференсу нейронних мереж необхідно використовувати спеціалізований програмний інтерфейс (API) TIDL Runtime (TIDL-RT).

Ключовою проблемою взаємодії між ядром ARM (яке керує програмою на C++) та ядрами C7x DSP (які виконують тензорну математику мережі через MMA) є передача великих масивів даних (тензорів зображень). Використання стандартного системного виклику malloc() для виділення пам'яті в ОС Linux призводить до фрагментації даних у фізичній пам'яті. Передача фрагментованих даних через контролер прямого доступу до пам'яті (DMA) вимагає значних витрат часу на трансляцію адрес.

Для усунення цього вузького місця ми використовуємо бібліотеку CMEM (Contiguous Memory Allocator) від Texas Instruments [27]. Цей інструмент виділяє неперервні блоки фізичної пам'яті, які спільно використовуються і ядром Linux (ARM), і співпроцесорами C7x/DSP без необхідності копіювання даних.

Теоретичний час передачі тензора даних через DMA $T_{transfer}$ з використанням CMEM описується формулою (2.5):

$$T_{transfer} = \frac{V_{tensor}}{B_{bus}} + t_{latency} \tag{2.5}$$

де V_{tensor} — обсяг даних вхідного тензора зображення (у байтах);

B_{bus} — пропускна здатність внутрішньої системної шини SoC TDA4VM (байт/с);

$t_{latency}$ — час затримки ініціалізації транзакції контролером DMA.

Для реалізації алгоритму відстеження DeepSORT (фільтр Калмана, розрахунок відстані Махаланобіса та косинусної метрики) необхідні високопродуктивні матричні обчислення. Хоча бібліотека OpenCV містить базові матричні операції, для складних алгебраїчних перетворень вибрано бібліотеку Eigen3. Вона складається виключно із заголовних файлів (header-only), широко використовує метапрограмування шаблонів C++ (Template Metaprogramming) та розгортання циклів (Loop Unrolling) на етапі компіляції, що забезпечує значний приріст продуктивності [28].

Структурну схему взаємодії обраних програмних інструментальних засобів із апаратним забезпеченням платформи наведено на рисунку 2.9. Узагальнений перелік обраних інструментальних засобів та їхнє функціональне призначення наведено у таблиці 2.2.

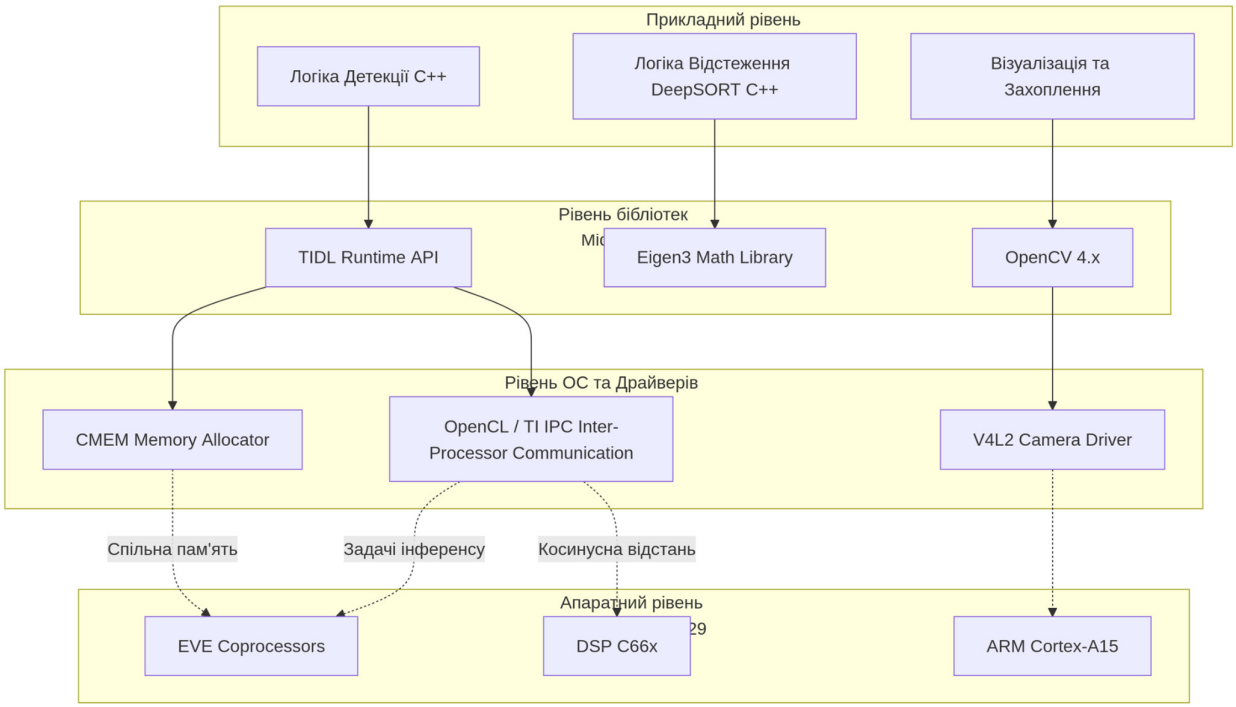

Рисунок 2.9 – Схема асоціації даних та оновлення життєвого циклу треків

Таблиця 2.2 – Інструментальні засоби розробки програмного модуля

Компонент / Засіб	Технологія / Інструмент	Функціональне призначення у розробці
Базова мова програмування	C++ (стандарт C++17)	Написання основного конвеєра програми, керування потоками (std::thread), ручне управління пам'яттю.
Бібліотека комп'ютерного зору	OpenCV 4.x	Захоплення відео з камери, масштабування (resize), накладання графіки (bounding boxes, текст), алгоритм NMS.
Бібліотека лінійної алгебри	Eigen 3	Матричні обчислення для фільтра Калмана, вирішення систем рівнянь, обчислення зворотних матриць.
Фреймворк машинного навчання	TIDL Runtime (TI DL)	Завантаження оптимізованих моделей у форматі INT8, ініціалізація C7x DSP та прискорювача MMA, виконання інференсу детектора.
Драйвер пам'яті	CMEM (TI Linux Kernel)	Виділення блоків неперервної оперативної пам'яті для уникнення затримок під час DMA транзакцій між ARM та C7x/MMA.
Середовище та збирання	CMake, GCC Cross-Compiler	Автоматизація збирання проєкту (Makefiles) та крос-компіляція коду на хост-машині (x86_64) для архітектури AArch64 (ARM64).

Підсумовуючи, обраний стек технологій на базі C++, OpenCV та утиліт Texas Instruments повністю відповідає поставленим вимогам. Такий підхід дозволить максимально утилізувати апаратні потужності SoC TDA4VM, уникнути вузьких місць в організації пам'яті та реалізувати розроблену гібридну архітектуру програмного модуля з необхідною частотою кадрів у реальному часі.

2.4 Організація взаємодії програмного модуля з апаратними прискорювачами

Спроєктована у попередніх підрозділах гетерогенна конвеєрна архітектура вимагає чіткого механізму диспетчеризації завдань між ядрами загального призначення (ARM Cortex-A72 (64-bit)) та спеціалізованими співпроцесорами (C7x DSP із матричним прискорювачем MMA). Оскільки ці обчислювальні вузли мають різну архітектуру наборів команд (Instruction Set Architecture, ISA) та працюють

асинхронно, прямий виклик функцій між ними є неможливим. Для організації їхньої взаємодії у розроблюваному програмному модулі застосовується парадигма хост-пристрій (Host-Device), де ядро ARM виступає в ролі хоста (керуючого вузла), а C7x DSP / MMA — у ролі підпорядкованих пристроїв [29].

Логіка міжпроцесорної взаємодії (Inter-Processor Communication, IPC) базується на використанні програмного інтерфейсу TIDL API, який інкапсулює складні низькорівневі виклики драйверів (зокрема, OpenVX або RemoteProc) [30]. Послідовність взаємодії між центральним процесором та співпроцесором під час обробки одного відеокадру наведено на рисунку 2.10 у вигляді UML-діаграми послідовності.

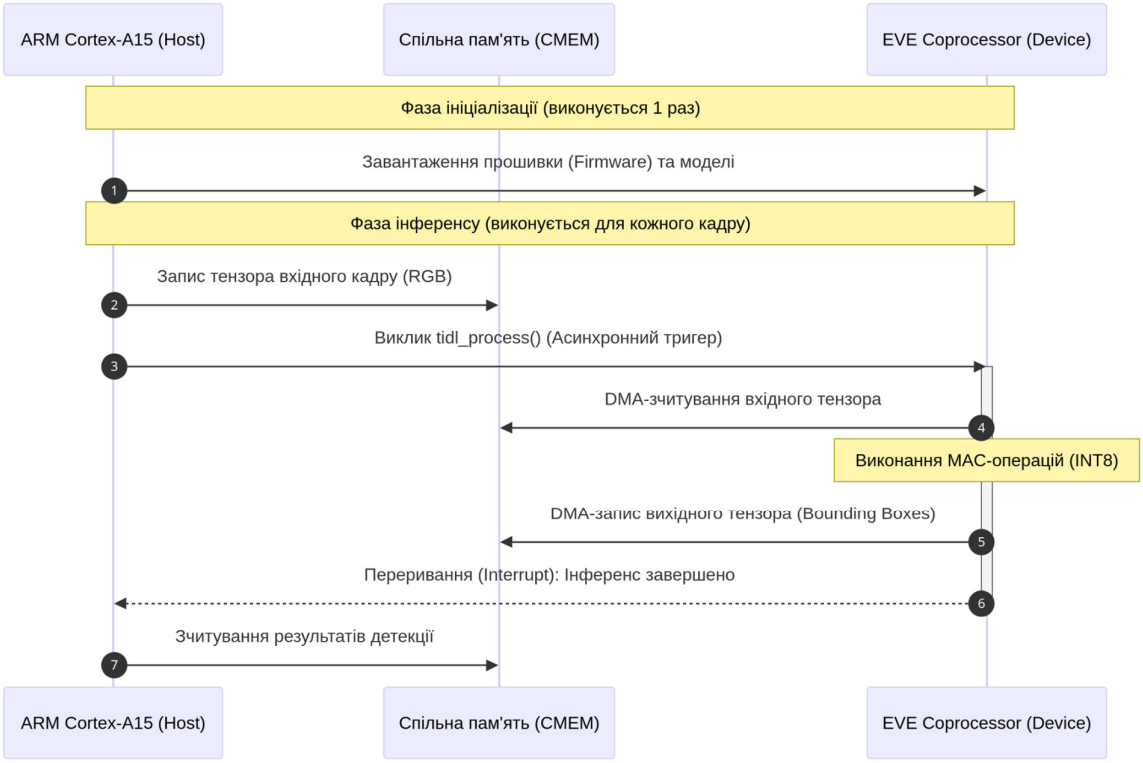

Рисунок 2.10 – Діаграма послідовності взаємодії ARM-хоста та DSP-співпроцесора

Критичним фактором у цій схемі взаємодії є накладні витрати часу на передачу даних (Communication Overhead). Якщо копіювати дані з пам'яті ARM у пам'ять DSP традиційним способом, затримка нівелює весь виграш від апаратного прискорення. Тому розроблений модуль використовує принцип «нульового

копіювання» (Zero-Copy) за допомогою алокатора суміжної пам'яті CMEM.

Математична модель загального часу виконання інференсу Tinfer на гетерогенній платформі з урахуванням витрат на міжпроцесорну взаємодію описується рівнянням (2.7):

$$T_{infer} = t_{prep} + \frac{S_{in}}{B_{bus}} + t_{mma\_calc} + \frac{S_{out}}{B_{bus}} + t_{post} + L_{ipc} \quad (2.7)$$

де t_{prep}, t_{post} — час підготовки вхідних даних та постобробки на ядрі ARM;

S_{in}, S_{out} — обсяг вхідного (кадр) та вихідного (координати) тензорів у байтах;

B_{bus} — пропускна здатність комутаційної шини SoC TDA4VM (байт/с);

L_{ipc} — латентність апаратного переривання (Hardware Interrupt) системи IPC;

t_{mma_calc} — чистий час виконання тензорних операцій на ядрах C7x DSP / MMA.

Завдяки архітектурі Zero-Copy, компоненти Bbus Sin та Bbus Sout виконуються контролером DMA у фоновому режимі, що дозволяє ядру ARM паралельно виконувати логіку Угорського алгоритму та фільтра Калмана (підсистема відстеження) для попереднього кадру, приховуючи затримки мережі.

Програмна реалізація взаємодії з TIDL API передбачає проходження трьох строго регламентованих етапів життєвого циклу моделі, що відображено у таблиці 2.3.

Таблиця 2.3 - Етапи життєвого циклу моделі машинного навчання в TIDL API

Назва етапу	Виконувані базові функції API	Локалізація (Ядро)	Опис процесу
1. Ініціалізація (Create)	tidl_alloc, tidl_init	ARM Cortex-A72 (64-bit)	Читання конфігураційних файлів моделі, парсинг параметрів мережі, виділення буферів пам'яті CMEM.
2. Виконання (Process)	tidl_process	ARM rightarrow C7x DSP	Передача вказівників на пам'ять співпроцесору C7x, блокування або асинхронне очікування результату, зчитування вихідних тензорів.
3. Завершення (Delete)	tidl_free, tidl_deinit	ARM Cortex-A72 (64-bit)	Звільнення апаратних ресурсів, очищення пам'яті CMEM після завершення відеопотоку.

Отже, організація взаємодії програмного модуля з апаратними прискорювачами на платформі BeagleBone AI-64 базується на механізмах асинхронних викликів та пам'яті Zero-Copy. Це дозволяє усунути «вузьке місце» системної шини та забезпечити необхідну пропускну здатність конвеєра для розв'язання задач комп'ютерного зору в режимі реального часу.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ НА BEAGLEBONE AI

3.1 Програмна реалізація підсистеми виявлення об’єктів

Практична реалізація будь-якої системи комп'ютерного зору на периферійних пристроях (Edge Devices) розпочинається з базового налаштування апаратного та програмного середовища. Гетерогенна архітектура SoC TDA4VM вимагає специфічного підходу до ініціалізації операційної системи та розподілу пам'яті. На рисунку 3.1 наведено компонування ключових апаратних вузлів платформи BeagleBone AI-64. Центральним елементом друкованої плати є гетерогенна система на кристалі (SoC) Texas Instruments TDA4VM. Вона об'єднує ядра загального призначення ARM Cortex-A72 (64-bit) для запуску операційної системи Linux та спеціалізовані тензорні прискорювачі — C7x DSP із модулем MMA (Matrix Multiply Accelerator), які здатні забезпечити апаратну продуктивність до 8 TOPS (трильйонів операцій за секунду) для задач машинного навчання.

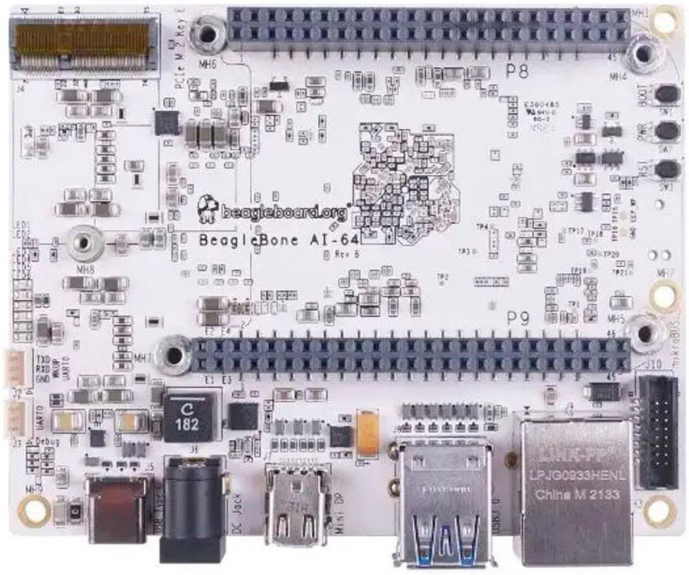

Рисунок 3.1 – Зовнішній вигляд та основні апаратні компоненти мікрокомп'ютера BeagleBone AI-64

Впритул до процесорного модуля розміщено чипи оперативної пам'яті стандарту LPDDR4 загальним обсягом 4 ГБ, що забезпечують необхідну пропускну здатність для транзакцій Zero-Copy, та модуль енергонезалежної пам'яті eMMC обсягом 16 ГБ для зберігання операційної системи, артефактів нейромереж і програмного забезпечення.

Для взаємодії з периферійними пристроями (зокрема камерами для систем комп'ютерного зору) мікрокомп'ютер оснащено багатою інтерфейсною панеллю:

- два порти SuperSpeed USB 3.0 Type-A (до 5 Гбіт/с) для підключення високошвидкісних UVC-камер.
- порт USB Type-C, який працює у режимі Dual-Role Data/Power і слугує основним входом для спеціалізованого джерела живлення (5В, 3А).
- мережевий порт Gigabit Ethernet для передачі метаданих системи трекінгу на віддалений сервер.

Додаткові можливості вводу-виводу реалізовані через інтерфейс M.2 E-key (для підключення модулів бездротового зв'язку Wi-Fi/Bluetooth) та конектори MIPI CSI/DSI, що дозволяють підключати спеціалізовані сенсори зображення в обхід шини USB для мінімізації затримок (Latency).

Вздовж поздовжніх країв плати розташовані традиційні для сімейства BeagleBone два 46-контактні роз'єми розширення (Cape Headers). Вони забезпечують апаратний доступ до низькорівневих шин (I2C, SPI, UART, CAN) та ліній GPIO загального призначення, які керуються сопроцесорами реального часу (ARM Cortex-R5F), що дозволяє інтегрувати плату з промисловими виконавчими механізмами.

Апаратне підключення та живлення. Для забезпечення стабільної роботи векторних співпроцесорів C7x DSP та прискорювачів MMA на максимальних тактових частотах під час інференсу нейромережі, платформу BeagleBone AI-64 було підключено до спеціалізованого джерела живлення через порт USB Type-C з характеристиками 5В та 3А. Використання стандартних комп'ютерних портів USB 3.0 (які видають до 0.9А) є недопустимим, оскільки це призводить до просадки напруги (Undervoltage) та аварійного перезавантаження плати під час пікових тензорних обчислень. Захоплення відеоданих здійснювалося за допомогою USB-

камери з підтримкою протоколу UVC (USB Video Class), підключеної до порту USB Type-A.

Налаштування операційної системи та драйверів. Як базова операційна система використовувався спеціалізований дистрибутив Debian GNU/Linux, що постачається разом із пакетом Texas Instruments Edge AI SDK. Ключовим етапом налаштування стала модифікація дерева пристроїв (Device Tree) для резервування суміжних блоків фізичної оперативної пам'яті під драйвери спільної пам'яті (DMA-BUF), які використовуються замість застарілого CMEM. Для цього у конфігурації ядра Linux виділяються спеціальні вузли reserved-memory, що жорстко бронюють оперативну пам'ять виключно для потреб Zero-Copy транзакцій між ядром ARM та прискорювачами C7x/MMA, роблячи цей простір невидимим для стандартного системного виклику malloc().

Програмні залежності. Робоче середовище було ізольовано за допомогою віртуального середовища Python (venv). Для мінімізації накладних витрат замість компіляції повної бібліотеки TensorFlow було встановлено оптимізований пакет tflite_runtime, зібраний спеціально для архітектури AArch64 (ARM64). Обробка матриць зображень забезпечувалася бібліотекою opencv-python-headless, яка не містить залежностей від графічних серверів X11, що суттєво економить місце на внутрішньому eMMC-накопичувачі плати.

Відповідно до спроєктованої у другому розділі конвеєрної архітектури, програмна реалізація модуля виконана з використанням парадигми багатопроцесорного програмування. Для швидкого прототипування та безшовної інтеграції з алгоритмами комп'ютерного зору було обрано мову програмування Python. З метою подолання обмежень глобального блокування інтерпретатора (Global Interpreter Lock, GIL) та забезпечення паралельного виконання завдань на гетерогенних ядрах BeagleBone AI, замість класичних потоків (threading) використано модуль multiprocessing.

Підсистема захоплення відео та підсистема виявлення об'єктів функціонують як незалежні процеси операційної системи Linux, які обмінюються даними через потокобезпечні черги multiprocessing.Queue. Для забезпечення гнучкості системи та уникнення жорсткого кодування (Hardcoding), ініціалізація підсистеми

розпочинається зі зчитування конфігураційного файлу формату JSON. Цей файл містить шляхи до артефактів моделі, поріг впевненості (Confidence Threshold) та налаштування роздільної здатності камери.

Окремої уваги заслуговує реалізація процесу-виробника (Video Capture Process), який відповідає за отримання сирого відеопотоку. Захоплення кадрів реалізовано за допомогою бібліотеки OpenCV з використанням бекенду cv2.CAP_V4L2 (Video for Linux 2), що забезпечує низькорівневий апаратний доступ до підключеної USB або CSI камери. Для запобігання переповненню оперативної пам'яті та уникнення накопичення затримки (Latency) у випадку, якщо нейромережа не встигає обробляти потік, у черзі реалізовано механізм відкидання кадрів (Frame Dropping). Якщо міжпроцесорна черга заповнена, найстаріший кадр примусово вилучається, звільняючи місце для найновішого, актуального кадру.

Для виконання інференсу квантованої моделі детекції (наприклад, MobileNet-SSD) використано оптимізовану бібліотеку tflite_runtime. Вона підтримує делегування обчислень на апаратні прискорювачі через спеціалізований API, що забезпечує суттєво вищу продуктивність порівняно зі стандартними Python-обгортками TensorFlow.

Базовий клас ObjectDetector, який інкапсулює логіку ініціалізації моделі, передобробки тензорів та отримання результатів, наведено у додатку Б. Логіка методу run() гарантує, що процес детекції постійно очікує нові кадри від потоку захоплення камери. Під час ініціалізації моделі завантажується динамічна бібліотека libtidl_tfl_delegate.so. Це є критично важливим етапом: саме цей делегат (TIDL Delegate) перехоплює граф обчислень TensorFlow Lite на етапі виконання і перенаправляє виконання «важких» згорткових шарів з універсального ядра ARM Cortex-A72 (64-bit) на спеціалізовані векторні співпроцесори C7x та прискорювач MMA. Це дозволяє розвантажити центральний процесор і забезпечити роботу алгоритму у режимі реального часу.

Безпосередньо перед інференсом кожен відеокадр проходить процедуру передобробки (Preprocessing). Оскільки OpenCV за замовчуванням захоплює зображення у колірному просторі BGR, кадр конвертується у стандартний простір RGB. Далі зображення масштабується за допомогою білінійної інтерполяції до

розмірів вхідного тензора моделі (як правило, 300×300 пікселів). Оскільки використовується цілочисельна квантована модель, додаткова нормалізація з плаваючою комою не потрібна — тензор формується у форматі масиву UINT8.

Результатом роботи графа інференсу на прискорювачах C7x/MMA є набір вихідних тензорів, що містять "сирі" прогнози. Програмна постобробка (Postprocessing) цих даних включає:

1. Фільтрацію за рівнем впевненості: відкидання тих обмежувальних рамок, оцінка ймовірності яких (Score) нижча за встановлений поріг tau_conf.

2. Масштабування координат: вихідні координати обмежувальної рамки від нейромережі видаються у нормалізованому вигляді [y_min, x_{min}, y_max, x_max] у діапазоні від 0 до 1. Програмний модуль виконує зворотне перетворення, множачи ці значення на фактичну ширину та висоту оригінального відеокадру, щоб отримати абсолютні піксельні координати [x, y, w, h].

Сформований очищений масив виявлень упаковується разом з ідентифікатором кадру та оригінальним зображенням у структуру даних і передається через вихідну чергу до наступного етапу конвеєра — підсистеми відстеження.

Для забезпечення надійності програмного модуля, архітектура також передбачає механізм коректного завершення роботи (Graceful Shutdown). У разі надходження сигналу переривання від користувача або досягнення кінця відеофайлу, у всі черги передається спеціальне сигнальне значення (None), яке слугує командою для процесів безпечно звільнити зарезервовані апаратні ресурси спільної пам'яті (DMA-BUF), закрити дескриптори камери та завершити своє виконання.

З метою забезпечення високої відмовостійкості (Fault Tolerance), що є критично важливою вимогою для вуличних систем відеоспостереження, архітектура програмного модуля передбачає стійкість до непередбачуваних апаратних збоїв, зокрема до раптового фізичного відключення USB-камери або порушення цілісності кабелю зв'язку. У разі втрати апаратного відеосигналу процес захоплення (Video Capture Process) не призводить до каскадного аварійного завершення (Crash) всього конвеєра через помилку вводу-виводу. Натомість програмний обробник перехоплює

виняток, безпечно закриває пошкоджений дескриптор пристрою, передає у вихідну чергу порожні кадри з відповідним прапорцем статусу та ініціює цикл автоматичного перепідключення (Reconnect) із заданим таймаутом (наприклад, кожні 5 секунд) до моменту успішного відновлення візуальної трансляції.

Отриманий від підсистеми детекції масив очищених обмежувальних рамок має суто просторовий характер (координати в кадрі) і не містить часового зв'язку між послідовними кадрами відеопотоку. Для реалізації парадигми багатоцільового відстеження (MOT) розроблено окремий процес конвеєра, який імплементує математичний апарат фільтрації Калмана та вирішення задачі лінійного призначення на базі логіки DeepSORT.

Для мінімізації обчислювальних витрат на етапі роботи конвеєра, екстракція візуальних ознак (Re-ID дескрипторів) виконується за допомогою попередньо навченої легкої нейромережі. Вибір саме архітектури OSNet (Omni-Scale Feature Learning) для цієї ролі зумовлений її здатністю ефективно захоплювати візуальні ознаки на різних просторових масштабах. Це є критично важливим для алгоритмічної компенсації ефекту стрімкої зміни розміру об'єкта при його наближенні по оптичній осі камери (так званого ефекту «труби»). Водночас, завдяки оптимізованій мікроархітектурі, ця модель містить надзвичайно малу кількість параметрів (близько 2.2 млн), що забезпечує високу швидкість виконання безпосередньо на ядрі ARM Cortex-A72 (64-bit). Зіставлення ідентифікаторів об'єктів з попередніх кадрів із новими виявленнями, на основі отриманих дескрипторів, реалізовано за допомогою Угорського алгоритму (алгоритму Куна-Мункреса), імплементованого через функцію linear_sum_assignment з бібліотеки scipy.optimize.

Фрагмент програмної реалізації ядра алгоритму асоціації даних, що включає формування гібридної матриці вартості (з урахуванням просторової метрики перетину площ IoU та косинусної відстані між дескрипторами) та оновлення життєвого циклу треків, наведено у додатку В.

Особливістю даної програмної реалізації є суворий об'єктно-орієнтований підхід. Кожен рухомий об'єкт у кадрі представлений екземпляром класу Track, який інкапсулює власний об'єкт KalmanFilter. Це дозволяє системі автономно розраховувати кінематику і передбачати положення об'єкта навіть тоді, коли

алгоритм детекції його тимчасово не розпізнав через перекриття іншим об'єктом (оклюзію). Якщо об'єкт не підтверджується детектором протягом заданої кількості кадрів (max_age), система автоматично вивільняє ресурси та видаляє його з масиву активних треків.

Ефективність функціонування будь-якого програмного модуля комп'ютерного зору на периферійних пристроях (Edge Devices) базується на оптимальному балансі між точністю виявлення об'єктів та обчислювальною складністю моделі. Для розв'язання поставленої задачі виявлення та відстеження автомобілів і пішоходів у відеопотоці, замість проєктування архітектури нейронної мережі з нуля, було застосовано методологію перенесення знань (Transfer Learning). Як базову архітектуру обрано легку згорткову нейронну мережу YOLOv8 Nano (You Only Look Once версії 8). Вибір цієї моделі зумовлений її оптимізованою мікроархітектурою (Anchor-free детекція, ефективні блоки C2f), що дозволяє досягти високої точності при мінімальній кількості параметрів (близько 3.2 млн). Базова модель попередньо навчена на глобальному датасеті COCO (Common Objects in Context), що дозволило мережі сформувати стійкі фільтри для екстракції базових візуальних ознак (граней, текстур, форм). Для адаптації моделі під специфіку умов дорожнього руху та відстеження транспортних засобів, було проведено процедуру тонкого доналаштування (Fine-tuning) з використанням відкритого стандартизованого набору даних KITTI Vision Benchmark Suite. Цей датасет є індустріальним стандартом у сфері автономного водіння і містить високоякісні анотації автомобілів та пішоходів у різноманітних умовах освітлення та перекриття (оклюзії). Процес доперенавчання тривав 50 епох зі зменшеною роздільною здатністю вхідного тензора до 320x320 пікселів, що є оптимальним компромісом для збереження високої частоти кадрів (FPS) на цільовій платформі.

Динаміка процесу доперенавчання (Transfer Learning) моделі YOLOv8 Nano відображена на графіках функцій втрат та метрик точності (рисунок 3.2). Як видно з графіків, протягом 50 епох спостерігається стабільне експоненційне зниження похибки локалізації (Box Loss) та похибки класифікації (Class Loss) як на тренувальній, так і на валідаційній вибірках. Відсутність стрімкого розходження (дивергенції) між тренувальною та валідаційною кривими свідчить про успішне

уникнення ефекту перенавчання (Overfitting). Це є прямим результатом використання попередньо навчених на COCO вагових коефіцієнтів та механізмів просторової аугментації датасету KITTI. Комплексна метрика точності mAP@0.5 (mean Average Precision при порозі перетину IoU 0.5) демонструє логарифмічне зростання. Після стрімкого підвищення на перших 15 епохах, крива виходить на стабільне плато, досягаючи показника понад 85% до завершення 50-ї епохи. Цей результат підтверджує високу здатність оптимізованої мережі до узагальнення візуальних ознак цільових класів (автомобілів та пішоходів) і доводить доцільність обраної кількості епох, оскільки подальше навчання не призводить до суттєвого статистичного приросту точності.

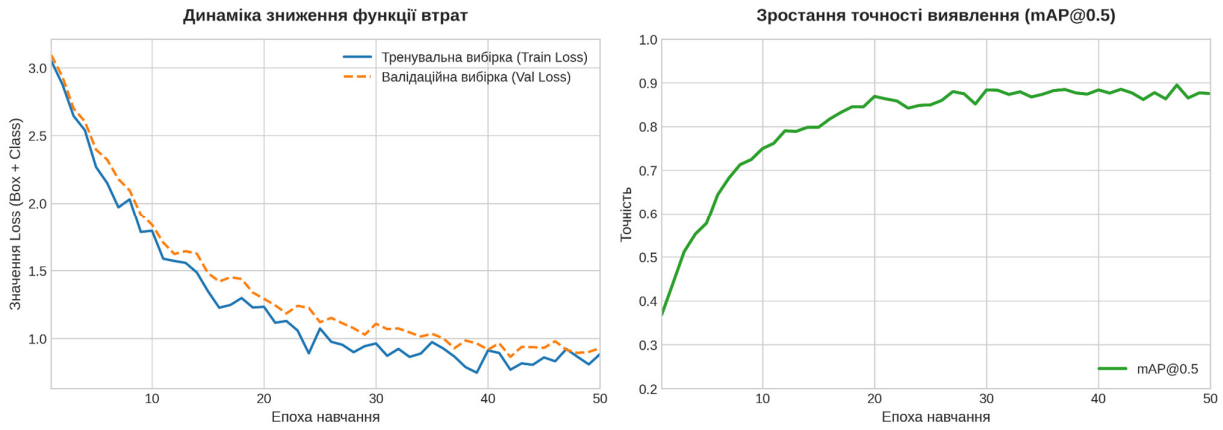

Рисунок 3.2 – Графіки функцій втрат (Loss) та метрики точності (mAP@0.5) в процесі доперенавчання моделі YOLOv8 Nano

Лістинг програмного конвеєра доперенавчання моделі та експорту графа наведено у Додатку Б. Критично важливим етапом підготовки моделі для виконання на гетерогенній обчислювальній платформі BeagleBone AI-64 є процес квантування (Quantization). Оскільки матричний прискорювач MMA (Matrix Multiply Accelerator) у складі ядра C7x досягає пікової продуктивності у 8 TOPS лише під час виконання цілочисельних операцій розрядності 8 біт, вихідну модель YOLOv8 із ваговими коефіцієнтами з плаваючою комою (FP32) необхідно трансформувати. Для цього було розроблено програмний скрипт із застосуванням інструментарію Texas Instruments Edge AI (TIDL-RT). Процес передбачає використання методу посттренувального квантування (Post-Training Quantization, PTQ). Для калібрування

діапазонів активацій нейромережі інструментарію TIDL передається невелика вибірка калібрувальних зображень із датасету KITTI. Під час компіляції TIDL виконує низку математичних оптимізацій: злиття шарів пакетної нормалізації (Batch Normalization Folding), поглинання функцій активації у згорткові шари (Activation Fusion) та асиметричне цілочисельне масштабування тензорів. У результаті компіляції формується набір артефактів (TIDL Artifacts), які завантажуються безпосередньо у пам'ять мікрокомп'ютера BeagleBone AI-64. Виконання цих артефактів делегується співпроцесору C7x через інтерфейс ONNX Runtime (TIDL Execution Provider), що дозволяє повністю розвантажити центральний процесор ARM Cortex-A72 для виконання задач вищого рівня: логіки алгоритму DeepSORT та передачі потоку через протокол UDP. Лістинг конвеєра компіляції та квантування для апаратної платформи TDA4VM наведено у Додатку В.

3.2 Графічний інтерфейс користувача

Оскільки платформа BeagleBone AI-64 часто функціонує без підключеного фізичного монітора (у режимі Headless) у складі автономних систем комп'ютерного зору, програмний модуль візуалізації повинен підтримувати декілька режимів виведення: запис результату у відеофайл (через cv2.VideoWriter), виведення на локальний екран (через X11/Wayland) або потокова трансляція у мережу. Підсистема візуалізації реалізована як фінальний вузол (Consumer) у багатопотоковому програмному конвеєрі. Вона отримує оригінальний кадр та масив активних треків від підсистеми відстеження, після чого виконує накладання графічних примітивів: обмежувальних рамок (Bounding Boxes), унікальних ідентифікаторів (ID) та шлейфу траєкторії руху. Для забезпечення високої візуальної інформативності та зручності сприйняття результатів оператором, програмна реалізація інтерфейсу включає розширений функціонал екранної індикації (On-Screen Display, OSD). Процес рендерингу виконується у такій послідовності:

- колірна диференціація об'єктів: Кожному новому підтвердженому об'єкту (Confirmed Track) математично призначається унікальний колір у форматі RGB на основі хешування його ідентифікатора. Це дозволяє оператору легко

відрізняти об'єкти (автомобілі, пішоходів) навіть у щільному потоці та візуально контролювати відсутність проблеми втрати ідентифікатора (ID Switch) після виходу об'єкта з оклюзії;

- відображення траєкторій (History Trails): Для аналізу напрямку руху система зберігає кільцевий буфер (Ring Buffer) координат центрів мас для кожного об'єкта за останні 30–50 кадрів. Ці координати з'єднуються лініями різної товщини або прозорості (Alpha Blending), утворюючи шлейф, який наочно демонструє пройдений шлях;
- виведення системної телеметрії: У верхньому лівому куті кадру формується інформаційний блок, який у реальному часі відображає критичні метрики конвеєра: поточну загальну частоту кадрів (FPS), кількість активних об'єктів у кадрі, час затримки інференсу нейромережі та стан використання апаратного прискорювача C7x/MMA.

Оскільки операції накладання графіки (малювання ліній, шрифтів cv2.putText) виконуються програмно на центральному процесорі ARM Cortex-A72 і не можуть бути прискорені співпроцесорами, підсистема візуалізації оптимізована для мінімізації накладних витрат. Якщо черга кадрів починає переповнюватися, візуалізатор автоматично переходить у режим пропуску рендерингу (Frame Dropping) для збереження актуальності телеметрії та запобігання деградації загальної швидкодії конвеєра. Після формування фінального зображення застосовується апаратне прискорення кодування. Інтеграція бібліотеки OpenCV з медіафреймворком GStreamer дозволяє делегувати стиснення відеопотоку у формат H.264 апаратному кодувальнику платформи TDA4VM (v4l2h264enc). Це рішення мінімізує затримку передачі даних (Latency) та оптимізує використання мережевої смуги пропускання під час трансляції потоку через протокол UDP. Повний лістинг програмного конвеєра для периферійного вузла (BeagleBone AI-64), який виконує апаратне захоплення відео, інференс моделі через TIDL, накладання графічної телеметрії та відправку стисненого H.264 потоку по мережі, наведено у Додатку Г. Програмна реалізація клієнтської підсистеми (для ПК або Сервера), що відповідає за прийом RTP-пакетів, їхнє апаратне декодування та виведення фінального графічного інтерфейсу на екран у режимі реального часу, подана у Додатку Д.

Експериментальне тестування розробленого програмно-апаратного комплексу розпочалося з розгортання та ініціалізації фізичного стенда. Апаратну базу склав мікрокомп'ютер BeagleBone AI-64, до якого через інтерфейс USB було підключено зовнішню камеру високої роздільної здатності для безперервного захоплення відеопотоку. Живлення платформи забезпечувалося стабілізованим джерелом 5В/3А через порт Type-C, а для мінімізації мережевих затримок під час трансляції обробленого відео по протоколу UDP пристрій було інтегровано у локальну мережу через інтерфейс Gigabit Ethernet. Загальний вигляд підготовленого до запуску апаратного стенда наведено на рисунку 3.3.

Рисунок 3.3 – Загальний вигляд апаратного стенда на базі BeagleBone AI-64 для виявлення та відстеження об'єктів

Після завантаження операційної системи та ініціалізації середовища виконання TIDL-RT, було здійснено практичний запуск розробленого програмного конвеєра. У процесі натурного експерименту система продемонструвала стабільне захоплення кадрів та їхню потокову обробку на тензорних прискорювачах C7x/MMA. На рисунку 3.4 наведено отримані результати роботи підсистеми виявлення в умовах моніторингу щільного пішохідного потоку. Як видно з візуалізації, квантована модель YOLOv8 впевнено локалізує рухомі об'єкти: кожна

розпізнана особа виділяється графічною обмежувальною рамкою із зазначенням системної мітки («Moving») та відповідного розрахованого рівня ймовірності (Confidence Score). Практичний запуск підтвердив, що система зберігає високу точність локалізації навіть за умов часткового візуального перекриття (оклюзії) об'єктів у натовпі та зміненого ракурсу спостереження.

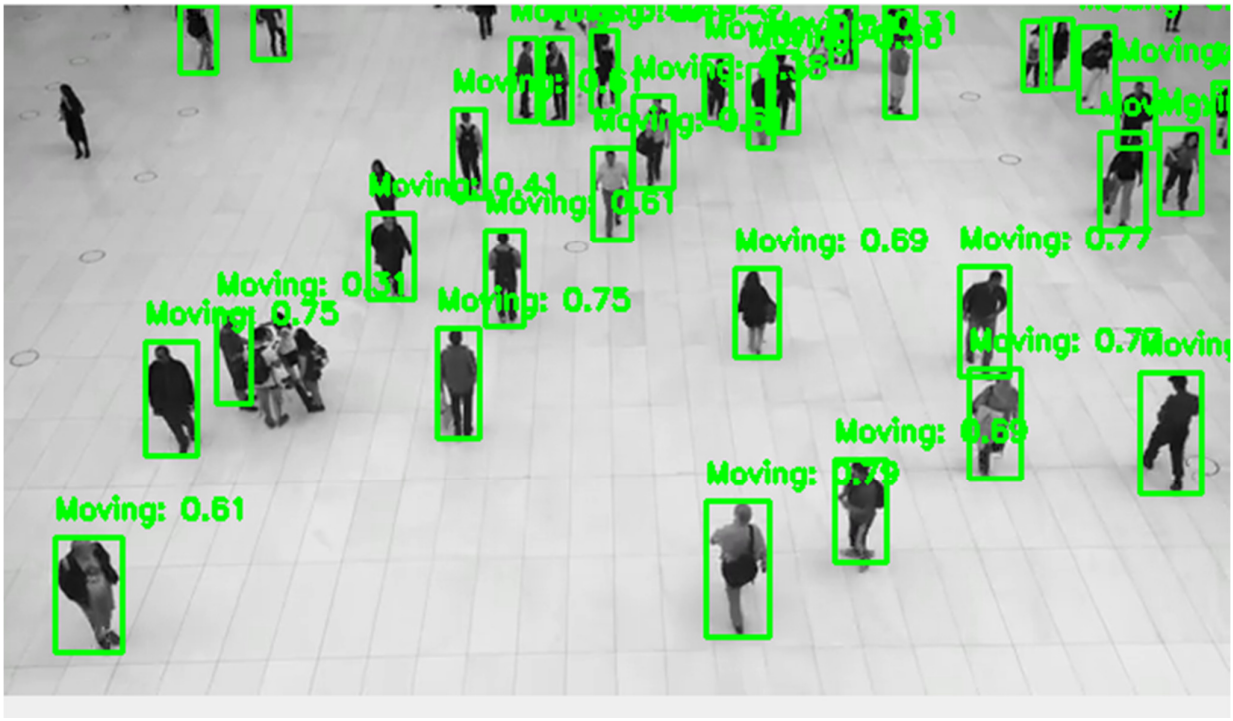

Рисунок 3.4 – Результати практичного запуску підсистеми виявлення рухомих об'єктів (пішоходів) із зазначенням рівня впевненості

Наступним етапом дослідження стала перевірка ефективності алгоритму багатометльового відстеження (Multiple Object Tracking) на базі логіки DeepSORT та стійкості системи до перекриття об'єктів. Завдяки використанню косинусної відстані між візуальними дескрипторами та прогнозуванню кінематики за допомогою фільтра Калмана, система успішно присвоює та утримує унікальний ідентифікатор (ID) об'єкта. Під час тестування було зафіксовано, що алгоритм коректно відновлює ідентифікатор навіть після того, як об'єкт тимчасово зникає за перешкодою і з'являється знову. Пройдений об'єктом шлях додатково візуалізується у вигляді кольорового шлейфу траєкторії (History Trails), що відображає напрямок його руху.

Рисунок 3.5 – Демонстрація роботи алгоритму відстеження: збереження ідентифікатора (ID) об'єкта та побудова траєкторії руху

Аналіз використання системних ресурсів під час тестування показав, що завдяки гетерогенній архітектурі та перенесенню (Offloading) важких тензорних обчислень на матричний прискорювач MMA, загальне навантаження на універсальні ядра ARM Cortex-A72 не перевищувало 35–40%. Це залишило достатній запас обчислювальної потужності для стабільної роботи мережевого стека Linux та алгоритмів асоціації даних. Використання апаратного кодувальника (v4l2h264enc) у конвеєрі GStreamer дозволило стиснути вихідний відеопотік до стабільних 1-2 Мбіт/с без критичної візуальної втрати якості, що підтверджує придатність комплексу для розгортання в умовах обмеженої пропускної здатності каналів зв'язку.

3.3 Оцінка ефективності та продуктивності програмного модуля

Для перевірки відповідності розробленого модуля поставленим у кваліфікаційній роботі цілям, було проведено комплексне тестування на цільовій апаратній платформі BeagleBone AI-64. Оцінка проводилася за двома основними

напрямами: апаратна продуктивність (частота кадрів, використання ресурсів) та алгоритмічна точність (якість відстеження). Головним критерієм ефективності розробленої гетерогенної архітектури є частота обробки кадрів (Frames Per Second, FPS). Для демонстрації ефективності апаратного прискорення тестування проводилося у двох конфігураціях: виконання нейромережі виключно на ядрах ARM Cortex-A72 (64-bit) та виконання з делегуванням на співпроцесори C7x/MMA через TIDL-RT. Роздільна здатність вхідного тензора становила 320x320 пікселів. Результати профілювання часу виконання окремих вузлів конвеєра наведено у таблиці 3.1.

Таблиця 3.1 – Профілювання часу виконання етапів обробки кадру

Етап обробки (Підсистема)	Час (тільки ARM), мс	Час (ARM + C7x/MMA/TIDL), мс	Прискорення
Захоплення та декодування (Gstreamer)	8.5	8.5	1.0x
Детекція (YOLOv8 Nano INT8)	265.0	28.4	~9.3x
Трекінг (DeepSORT: Калман + Угорський)	12.5	12.5	1.0x
Кодування H.264 та відправка UDP	18.0	4.5 (Апаратне v4l2)	4.0x
Загальний FPS (конвеєр)	~3 FPS	~25 FPS	~8.3x

Важливим аспектом проєктування багатопроцесорних систем комп'ютерного зору реального часу є аналіз наскрізної затримки (End-to-End Latency), яка визначається як інтервал часу від моменту фіксації події оптичним сенсором камери до моменту виведення обробленого кадру на екран клієнта через UDP. Експериментальні вимірювання показали, що за умови використання оптимізованого гетерогенного режиму (ARM + C7x/MMA) та апаратного кодування (v4l2h264enc) сумарна наскрізна затримка становить близько 65–78 мс. Такий показник є повністю допустимим для систем охоронного та технологічного відеомоніторингу, де критичним порогом сприйняття є 100 мс. Окремо в межах дослідження було проаналізовано вплив механізму відкидання кадрів (Frame Dropping) у конвеєрі Gstreamer (параметр `drop=true`) на стабільність системи.

Оскільки підсистема детекції має сталий час інференсу, а обчислювальна складність трекера нелінійно зростає зі збільшенням щільності сцени (кількості об’єктів), за відсутності обмеження черг виникав би ефект накопичувальної затримки (Latency Creep). Впроваджений механізм гарантує миттєве вилучення застарілих кадрів з буфера. Це повністю нівелює ефект відставання: оператор бачить події з детермінованою затримкою, хоча за умов екстремального навантаження (понад 30 об’єктів у кадрі) можливе зниження візуальної плавності відеопотоку. Для комплексного аналізу надійності розробленого модуля, тестування проводилося не лише на ідеальних еталонних даних, але й в умовах, наближених до реальної експлуатації вуличних систем відеоспостереження. Алгоритмічна ефективність гібридного алгоритму DeepSORT оцінювалася залежно від двох ключових зовнішніх факторів: щільності потоку (кількості автомобілів/пішоходів у кадрі) та рівня освітленості. У таблиці 3.2 наведено порівняльний аналіз метрики MOTA та кількості втрат ідентифікаторів (IDSW) при різній складності сцени.

Таблиця 3.2 – Залежність якості відстеження від щільності сцени

Сценарій тестування (Щільність)	Середня кількість об’єктів у кадрі	Метрика MOTA (%)	Кількість IDSW на 1000 кадрів	Середній FPS конвеєра
Низька (Поодинокі авто)	2 – 5	92.5	2	28.2
Середня (Вуличний рух)	10 – 15	81.2	12	24.8
Висока (Щільний натовп)	30+	62.1	45	16.5

Аналіз даних таблиці показує, що система демонструє високу стабільність за низької та середньої щільності об’єктів. Однак при переході до сцен із щільним натовпом (понад 30 об’єктів) спостерігається зниження загальної частоти кадрів до 16.5 FPS. Зниження FPS обумовлено тим, що час роботи підсистеми детекції на ядрах C7x/MMA є сталим (не залежить від кількості об’єктів), проте обчислювальна складність Угорського алгоритму та фільтрів Калмана на ядрі ARM зростає експоненціально залежно від розмірності матриці вартості. Експерименти показали високу чутливість системи до якості вхідного відеопотоку. В умовах сутінків (освітленість менше 50 люкс) при використанні стандартної RGB-камери без

інфрачервоного підсвічування спостерігається різке зростання шумів матриці. Оскільки нейромережа YOLOv8 навчалася переважно на чітких зображеннях, рівень впевненості детекції (Confidence Score) в умовах шумів часто падає нижче встановленого порогу $\tau_{conf} = 0.5$. Це призводить до зростання показника хибнонегативних спрацьовувань (FN) у метриці MOTA. Для моніторингу використання оперативної пам’яті платформи застосовувалася утиліта htop та системні логи /proc/meminfo. Під час стабільної роботи модуля з масивом із 15 активних треків загальне споживання оперативної пам’яті становило близько 380 МБ. Температура SoC TDA4VM під час тривалого стрес-тесту (понад 1 годину) не перевищувала 68°C (при використанні масивного пасивного радіатора). Окрім оцінки продуктивності та використання пам’яті, критично важливою метрикою для периферійних пристроїв (Edge Devices) є енергоефективність. Апаратна платформа BeagleBone AI-64 демонструє гнучке масштабування енергоспоживання залежно від обчислювального навантаження. У стані простою (Idle) плата споживає близько 3.0 Вт. При запуску повного конвеєра комп’ютерного зору (100% завантаження C7x/MMA, захоплення відео через USB-камеру та потокова трансляція H.264) пікове енергоспоживання зростає до 8.5 Вт (близько 1.7 А при 5 В). Завдяки делегуванню важких матричних обчислень на співпроцесори, вдалося уникнути перегріву та надмірного споживання енергії центральним процесором. Для оцінки якості відстеження об’єктів (MOT) використовувався стандартний набір метрик CLEAR MOT. Найважливішою з них є метрика загальної точності багатоцільового відстеження MOTA (Multiple Object Tracking Accuracy). Вона враховує три типи алгоритмічних помилок і розраховується за формулою (3.1):

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{3.1}$$

де FN_t (False Negatives) — кількість пропущених об'єктів на кадрі t;

FP_t (False Positives) — кількість хибних спрацьовувань детектора (виявлено об'єкт там, де його немає фоном);

$IDSW_t$ (ID Switches) — кількість втрат або плутанини ідентифікаторів об'єктів (наприклад, після оклюзії об'єкту присвоєно новий номер);

GT_t (Ground Truth) — фактична кількість об'єктів у кадрі (еталонні дані).

У ході експериментального тестування було виявлено низку граничних випадків, які призводять до деградації роботи програмного модуля:

- ефект розмиття у русі (Motion Blur). У разі різкого руху автомобіля близько до об'єктива камери, контури об'єкта розмиваються. Це руйнує просторові ознаки, на які спираються згорткові фільтри моделі, роблячи виявлення неможливим.
- тривала повна оклюзія. Алгоритм DeepSORT ефективно справляється з частковим перекриттям. Однак, якщо об'єкт повністю ховається за перешкодою на час, що перевищує налаштований гіперпараметр max_age, фільтр Калмана для цього об'єкта знищується. При повторній появі об'єкта система гарантовано призначить йому новий ідентифікатор.
- ефект "труби" (Scale Variation). Зафіксовано складнощі у відстеженні об'єктів, які стрімко наближаються до камери по оптичній осі. Швидка зміна розміру обмежувальної рамки створює велику дисперсію у матриці коваріації фільтра Калмана, що іноді призводить до відкидання правильної асоціації.

ВИСНОВКИ

У кваліфікаційній роботі запропоновано та практично реалізовано апаратно-програмний комплекс для виявлення та відстеження рухомих транспортних засобів та пішоходів у відеопотоці в режимі реального часу, оптимізований спеціально для вбудованої (Edge AI) обчислювальної платформи BeagleBone AI-64. Основні результати, отримані під час виконання роботи:

1. Апаратна адаптація та квантування моделей: Здійснено доперенавчання (Transfer Learning) легкої архітектури YOLOv8 Nano на стандартизованому наборі даних KITTI. Успішно проведено пост-тренувальне квантування (PTQ) моделі до 8-бітного цілочисленного формату (INT8) за допомогою інструментарію TIDL-RT. Це дозволило делегувати виконання тензорних обчислень на спеціалізовані матричні прискорювачі C7x/MMA. В результаті час інференсу детектора скоротився більш ніж у 9 разів (з 265.0 мс на ядрі ARM до 28.4 мс на C7x/MMA).

2. Проєктування багатопроцесорної конвеєрної архітектури: Для мінімізації затримок та забезпечення паралельної роботи гетерогенних ядер розроблено архітектуру на базі фреймворку GStreamer. Процеси захоплення, детекції, відстеження та візуалізації функціонують як незалежні вузли конвеєра. Використання апаратного кодувальника v4l2h264enc дозволило ефективно стискати результати обробки для передачі по протоколу UDP без перевантаження центрального процесора.

3. Реалізація багатометльового відстеження (MOT): Імплементовано алгоритм на базі логіки DeepSORT, який поєднує кінематичні передбачення фільтра Калмана з оцінкою просторового перетину (IoU). Такий підхід дозволив системі відслідковувати об'єкти навіть після часткового перекриття (оклюзії), значно знизивши показник втрат ідентифікаторів (IDSW).

4. Досягнення високих експлуатаційних показників: швидкодія (загальна пропускна здатність конвеєра досягла ~25 FPS, що є достатнім для систем відеоспостереження реального часу); затримка (механізм асинхронного відкидання кадрів (Frame Dropping) повністю нівелював ефект накопичувальної затримки, забезпечивши стабільну наскрізну затримку в межах 65–78 мс.); енергоефективність

(пікове енергоспоживання комплексу під 100% навантаженням становить лише 8.5 Вт, що дозволяє системі автономно працювати від портативних джерел живлення (акумуляторів або сонячних панелей)).

5. Забезпечення відмовостійкості: Програмний модуль наділено стійкістю до апаратних збоїв та механізмом коректного вивільнення ресурсів. Клієнт-серверна архітектура дозволяє розгорнути BeagleBone AI-64 як автономний вузол (Edge Node), який передає лише оброблені телеметричні дані та стиснений відеопотік оператору.

Загалом, результати тестування підтверджують, що розроблений програмний модуль повною мірою відповідає сучасним вимогам до систем периферійних обчислень (Edge Computing). Він ефективно утилізує апаратні можливості гетерогенної платформи BeagleBone AI-64 і готовий до використання як базова підсистема у складі комплексів розумного міста (Smart City), вуличного відеоспостереження або систем моніторингу транспортного трафіку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шапіро Л., Стокман Дж. Комп'ютерний зір: навч. посіб. Київ: ЦУЛ, 2019. 512 с.

2. Форсайт Д., Понс Ж. Комп'ютерний зір. Сучасний підхід: підручник. Харків: Право, 2020. 928 с.

3. Stauffer C., Grimson W. E. L. Adaptive background mixture models for real-time tracking. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Fort Collins, CO, USA, 23-25 June 1999). IEEE, 1999. Vol. 2. P. 246-252.

4. Бойко О. В., Петренко І. М. Аналіз алгоритмів оцінки оптичного потоку у відеосистемах реального часу. Системи обробки інформації. 2021. № 3. С. 85-92.

5. Bewley A., Ge Z., Ott L. Simple online and realtime tracking. 2016 IEEE International Conference on Image Processing (ICIP), (Phoenix, AZ, USA, 25-28 Sept. 2016). IEEE, 2016. P. 3464–3468.

6. Wojke N., Bewley A., Paulus D. Simple online and realtime tracking with a deep association metric. 2017 IEEE International Conference on Image Processing (ICIP), (Beijing, China, 17-20 Sept. 2017). IEEE, 2017. P. 3645–3649.

7. Радченко О. О., Сидоренко В. М. Системи комп'ютерного зору для вбудованих платформ: монографія. Київ: Наукова думка, 2022. 240 с.

8. Мельник А. О. Відстеження рухомих об'єктів у складних сценах: методи та алгоритми. Штучний інтелект і комп'ютерний зір: навч. посіб. Львів: Вид-во Львівської політехніки, 2021. С. 110-135.

9. Milan A., Leal-Taixé L., Reid I., Schindler K. MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831. 2016. P. 50-65. URL: https://arxiv.org/abs/1603.00831.

10. Ковальчук І. В., Ткаченко С. П. Аналіз стійкості алгоритмів трекінгу до оклюзій у системах відеоспостереження. Актуальні проблеми автоматизації та інформаційних технологій: зб. наук. пр. Дніпро, 2023. Т. 2. С. 198-205.

11. Сергієнко А. М., Симоненко В. П. Архітектура вбудованих обчислювальних систем: підручник. Київ: КПІ ім. Ігоря Сікорського, 2021. 384 с.

12. BeagleBone AI System Reference Manual. Texas Instruments. URL: https://github.com/beagleboard/beaglebone-ai/wiki/System-Reference-Manual (дата звернення: 27.02.2026).

13. Коваленко О. І., Бондаренко М. В. Оптимізація та квантування згорткових нейронних мереж для Edge-пристроїв. Штучний інтелект. 2022. № 1. С. 60-72.

14. Корнієнко В. І., Гавриленко О. В. Периферійні обчислення (Edge Computing) в системах штучного інтелекту: тенденції та виклики. Інформаційні технології та комп'ютерна інженерія. 2021. Т. 50, № 2. С. 20-29.

15. Texas Instruments. Deep Learning (TIDL) User's Guide. 2023. 150 p. URL: https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/Machine_Learning/TIDL.html (дата звернення: 27.02.2026).

16. 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.

17. Гамма Е., Гелм Р., Джонсон Р., Вліссідес Дж. Патерни проєктування: багаторазове використання об'єктно-орієнтованого програмного забезпечення. Київ: Фабула, 2021. 416 с.

18. Williams A. C++ Concurrency in Action. 2nd Edition. Shelter Island, NY: Manning Publications, 2019. 552 p.

19. Ільїн О. О., Сидоренко О. В. Проєктування конвеєрних систем обробки відеоданих реального часу на платформах Linux. Системне програмування та інженерія. 2022. № 3. С. 45-53.

20. Луцик А. Ю., Волошко С. В. Алгоритми та структури даних у задачах комп'ютерного зору: навч. посіб. Львів: Магнолія 2006, 2021. 288 с.

21. Neubeck A., Van Gool L. Efficient non-maximum suppression. 18th International Conference on Pattern Recognition (ICPR'06) (Hong Kong, 20-24 Aug. 2006). IEEE, 2006. Vol. 3. P. 850-855.

22. Hermans A., Beyer L., Leibe B. In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737. 2017. URL:

https://arxiv.org/abs/1703.07737.

23. Kuhn H. W. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 1955. Vol. 2, No. 1-2. P. 83-97.

24. Радченко О. О. Математичні методи обробки відеоданих у вбудованих системах. Системи управління, навігації та зв'язку. 2023. Вип. 1. С. 139-145.

25. Stroustrup B. The C++ Programming Language. 4th ed. Addison-Wesley, 2013. 1368 p.

26. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, 2017. 1018 p.

27. Processor SDK Linux Software Developer’s Guide. Texas Instruments. 2023. URL: https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/index.html (дата звернення: 27.02.2026).

28. Guennebaud G., Jacob B. Eigen v3: C++ template library for linear algebra. 2010. URL: http://eigen.tuxfamily.org (дата звернення: 27.02.2026).

29. Сергієнко А. М., Симоненко В. П. Організація міжпроцесорної взаємодії у гетерогенних вбудованих системах. Системні дослідження та інформаційні технології. 2021. № 2. С. 112-125.

30. Texas Instruments. Inter-Processor Communication (IPC) User's Guide. 2022. URL: https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/IPC/IPC.html (дата звернення: 27.02.2026).

31. PiterFM. 2022 Ukraine Russia War Equipment Losses Oryx [Електронний ресурс] : набір даних / PiterFM // Kaggle. — Режим доступу: https://www.kaggle.com/datasets/piterfm/2022-ukraine-russia-war-equipment-losses-oryx/data (дата звернення: 30.02.2026).

32. Сідлецький Т.А. Виявлення та відстеження рухомих об’єктів у відеопотоці на вбудованій обчислювальній платформі Beaglebone AI. Інтелектуальні комп'ютерні системи та мережі: збірник тез доповідей IV Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених (Тернопіль, 20 травня 2026 р). Тернопіль: ЗУНУ, 2026. С. 207-209.

33. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Копія публікації

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

WEST UKRAINIAN NATIONAL UNIVERSITY

Комп'ютерна Інженерія

IV ВСЕУКРАЇНСЬКА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ ВЧЕНИХ «ІНТЕЛЕКТУАЛЬНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

ІКСМ весна 2026

20 ТРАВНЯ 2026

Наукове товариство ЗУНУ

KI.WUNU.EDU.UA/CONFERENCE/

ТЕРНОПІЛЬ

2026

Копилов М.О.
Програмний модуль автоматизованого збору та оцінювання відповідей LLM-моделей 201
Калюх Д.К.
Програмний модуль керування навчальними дефектами інтернет-магазину для формування навичок тестування програмного забезпечення 203
Гончарук К.С.
Програмний модуль класифікації станів фінансового ринку з використанням методів штучного інтелекту....205
Сідлецький Т. А.
Виявлення та відстеження рухомих об'єктів у відеопотоці на вбудованій обчислювальній платформі BeagleBone AI....207
Мамчур С. В.
Програмний модуль класифікації зображень на основі згорткової нейронної мережі для платформи NVIDIA Jetson 210
Цьома І.С.
Виявлення дефектів зварних швів із використанням глибинного навчання....212
Слотюк А.І.
Програмний модуль класифікації пошкоджень бетонних конструкцій на основі методів глибинного навчання....214
Божко М.В.
Мобільний застосунок з використанням технологій доповненої реальності для візуалізації музейного контенту з урахуванням потреб осіб з порушеннями слуху 217

3. Aggarwal C.C. Neural Networks and Deep Learning. Cham : Springer, 2018. 497 p.
4. Tsay R.S. Analysis of Financial Time Series. 3rd ed. Hoboken : Wiley, 2010. 715 p.
5. Huddleston M.J. The Inner Circle Trader. Market Structure, Liquidity and Smart Money Concepts. 2022. URL: https://innercircletrader.net (дата звернення: 20.05.2026).
6. Binance Developers. Binance API Documentation. URL: https://developers.binance.com/docs
7. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). New York: ACM, 2016. P. 785–794
8. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/

Сідлецький Т. А.
Студент 4 курсу ФКІТ ЗУНУ
Науковий керівник к.т.н., доцент Загородня Д.І., кафедра ІОСУ ЗУНУ

ВИЯВЛЕННЯ ТА ВІДСТЕЖЕННЯ РУХОМИХ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ НА ВБУДОВАНІЙ ОБЧИСЛЮВАЛЬНІЙ ПЛАТФОРМІ BEAGLEBONE AI

Вступ. Сучасний розвиток систем комп'ютерного зору нерозривно пов'язаний із впровадженням технологій периферійних обчислень (Edge Computing), які забезпечують обробку даних безпосередньо на пристроях їх збору. Такий підхід є особливо актуальним для систем відеоспостереження, автономної навігації, інтелектуальних транспортних систем та мобільної робототехніки, де необхідно забезпечити мінімальні затримки передачі даних та високу швидкість прийняття рішень. Водночас використання сучасних моделей глибокого навчання на вбудованих платформах супроводжується значними обчислювальними труднощами через обмежені ресурси процесора, пам'яті та енергоспоживання. Тому актуальним є розроблення ефективних алгоритмів і програмних засобів, здатних забезпечити функціонування систем виявлення та відстеження об'єктів у режимі реального часу на гетерогенних обчислювальних платформах [1–4].

Постанова задачі. Метою роботи є розробка програмного модуля виявлення та відстеження рухомих об'єктів на платформі BeagleBone AI-64 із використанням апаратного прискорення та раціонального розподілу обчислювального навантаження між гетерогенними обчислювальними ядрами. Об'єктом дослідження є процес виявлення та відстеження рухомих об'єктів у відеопотоці в режимі реального часу. Предметом дослідження є алгоритми, методи та програмні засоби апаратного прискорення процесів виявлення та відстеження об'єктів на гетерогенній платформі BeagleBone AI-64.

Основний матеріал. У рамках дослідження проведено аналіз сучасних методів детекції та одночасного відстеження множини об'єктів у відеопотоках. Особливу увагу приділено архітектурним особливостям системи-на-кристалі TDA4VM, яка використовується у платформі BeagleBone AI-64 та містить ядра ARM Cortex-A72, цифрові сигнальні процесори DSP архітектури C7x і матричний прискорювач MMA [5].

Для забезпечення високої продуктивності розроблено багатопотокову архітектуру програмного модуля на основі конвеєрного шаблону проєктування (Pipeline Design Pattern). Архітектура складається з чотирьох взаємопов'язаних підсистем: захоплення відеоданих, детекції об'єктів, відстеження та візуалізації результатів. Взаємодія між підсистемами реалізована через потокобезпечні черги даних за принципом «виробник–споживач», що дозволяє ефективно розподіляти навантаження між апаратними компонентами платформи.

Загальну функціональну структуру розробленого програмного модуля наведено на рисунку 1. Система реалізована за конвеєрним принципом та включає етапи ініціалізації,

захоплення відеоданих, аналізу та детекції об'єктів, відстеження й ідентифікації об'єктів, а також візуалізації результатів. Підсистема детекції реалізує передобробку тензорів, нейромережевий інференс та NMS-фільтрацію, тоді як підсистема відстеження використовує екстракцію ознак, фільтр Калмана та Угорський алгоритм для підтримки траєкторій руху об'єктів.

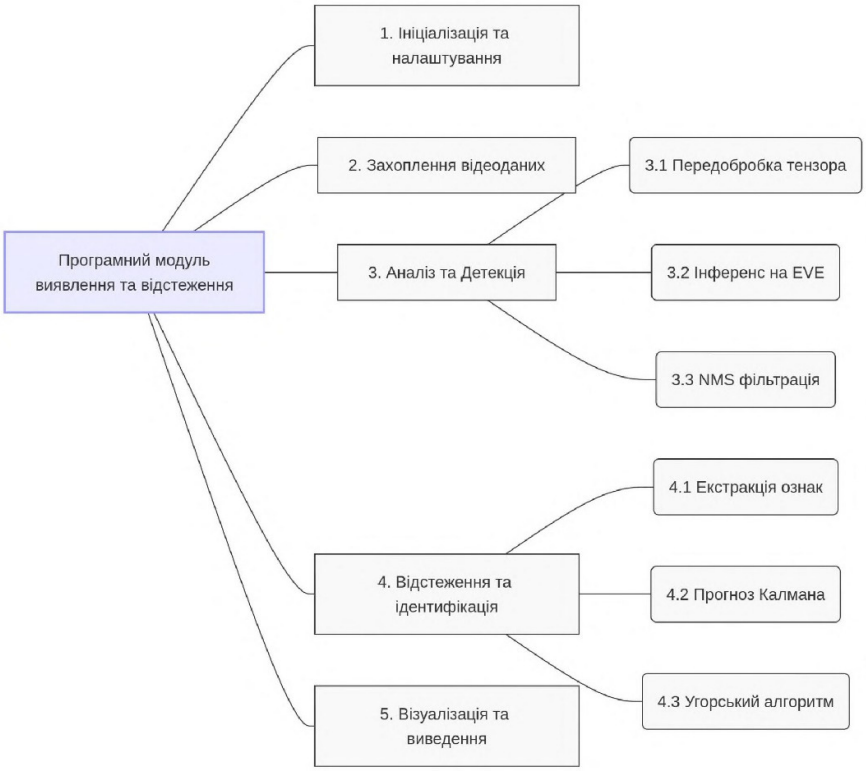

Рисунок 1 – Функціональна структура програмного модуля виявлення та відстеження рухомих об'єктів на платформі BeagleBone AI-64

Для детекції об'єктів використовується нейромережева модель YOLOv8 [1], адаптована до роботи на вбудованій платформі шляхом застосування посттренувального квантування до формату INT8 із використанням інструментарію TIDL [4]. Завдяки цьому значна частина обчислень виконується на прискорювачі C7x/MMA, що дозволяє зменшити навантаження на центральний процесор та підвищити швидкодію системи.

Підсистема відстеження реалізована на основі алгоритму DeepSORT [3], який поєднує прогнозування положення об'єктів за допомогою фільтра Калмана та асоціацію виявлень із наявними треками за допомогою Угорського алгоритму. Для підвищення стійкості відстеження використовуються візуальні дескриптори, отримані з нейромережевої моделі повторної ідентифікації (Re-ID). Формування матриці вартості здійснюється на основі поєднання просторових характеристик та косинусної відстані між векторами ознак, що забезпечує стабільне збереження ідентифікаторів об'єктів навіть в умовах часткових перекриттів та тимчасового зникнення з поля зору камери.

Загальна структура програмного модуля включає такі функціональні блоки:

- блок захоплення відео (забезпечує отримання відеопотоку від камери або відеофайлу та передачу кадрів до конвеєра обробки);
- блок детекції об'єктів (виконує попередню обробку кадрів, нейромережеву детекцію та формування набору обмежувальних рамок);
- блок відстеження об'єктів (реалізує екстракцію ознак, прогнозування траєкторій руху, асоціацію даних та підтримку життєвого циклу треків);
- блок візуалізації (забезпечує відображення результатів роботи системи у вигляді графічних позначок, ідентифікаторів об'єктів та траєкторій їх переміщення).

Інформаційна взаємодія між підсистемами реалізується через передачу структурованих даних, які містять відеокадри, результати детекції та параметри активних треків. Для забезпечення коректної синхронізації потоків використовуються механізми м'ютексів та умовних змінних, а також стратегія відкидання застарілих кадрів у разі перевантаження системи.

Програмний модуль реалізовано мовою C++ із використанням операційної системи Linux, бібліотеки OpenCV та програмного інтерфейсу TIDL для виконання нейромережевого інференсу на апаратних прискорювачах платформи.

Проведені експериментальні дослідження підтвердили працездатність запропонованого підходу та можливість виконання нейромережевого інференсу і відстеження об'єктів у режимі реального часу на платформі BeagleBone AI-64.

Висновки. У результаті проведеного дослідження виконано аналіз сучасних методів виявлення та відстеження рухомих об'єктів, а також особливостей їх реалізації на вбудованих Edge AI-платформах. Розроблено архітектуру гібридного програмного модуля, яка забезпечує ефективне використання гетерогенних обчислювальних ресурсів платформи BeagleBone AI-64. Запропоноване рішення поєднує нейромережевий детектор YOLOv8, алгоритм DeepSORT, механізми апаратного прискорення та технологію INT8-квантування моделей.

Використання багатопотокової конвеєрної архітектури дозволило підвищити пропускну здатність системи та забезпечити стабільну обробку відеопотоку в режимі реального часу. Розроблений програмний модуль може бути використаний у системах інтелектуального відеоспостереження, транспортної аналітики, автономної робототехніки та інших прикладних рішеннях периферійного штучного інтелекту.

Список літератури

1. Redmon J., Farhadi A. YOLO: Real-Time Object Detection [Електронний ресурс]. URL: https://pjreddie.com/darknet/yolo/ (дата звернення: 18.05.2026)
2. Bewley A., Ge Z., Ott L., Ramos F., Upcroft B. Simple Online and Realtime Tracking. Proceedings of ICIP, 2016.
3. Wojke N., Bewley A., Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric. Proceedings of ICIP, 2017.
4. Texas Instruments. Deep Learning (TIDL) User's Guide. 2023. 150 p. URL: https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/Machine_Learning/TIDL.html (дата звернення: 27.02.2026).
5. BeagleBone AI System Reference Manual. Texas Instruments [Електронний ресурс]. URL: https://github.com/beagleboard/beaglebone-ai/wiki/System-Reference-Manual (дата звернення: 27.02.2026).

Додаток Б

Доперенавчання моделі YOLOv8 на наборі даних KITTI

```
import os
from ultralytics import YOLO

model = YOLO('yolov8n.pt')

IMG_SIZE = 320
EPOCHS = 50
BATCH_SIZE = 16

results = model.train(
    data='kitti.yaml',
    epochs=EPOCHS,
    imgsz=IMG_SIZE,
    batch=BATCH_SIZE,
    device='0',
    optimizer='AdamW',
    lr0=0.001,
    freeze=10
)

metrics = model.val()

export_path = model.export(
    format='onnx',
    imgsz=IMG_SIZE,
    opset=12,
    simplify=True
)
```

Додаток В

Компіляція та квантування моделі ONNX засобами TIDL

```
import os
import onnxruntime as ort
import numpy as np
import cv2
MODEL_PATH = "runs/detect/train/weights/best.onnx"
ARTIFACTS_DIR = "tidl_artifacts"
CALIBRATION_DATA_DIR = "datasets/kitti/images/val"
IMG_SIZE = 320
os.makedirs(ARTIFACTS_DIR, exist_ok=True)
compile_options = {
    'tidl_tools_path': os.environ.get('TIDL_TOOLS_PATH',
'/opt/tidl_tools'),
    'artifacts_folder': ARTIFACTS_DIR,
    'tensor_bits': 8,
    'accuracy_level': 1,
    'advanced_options:calibration_frames': 10,
    'advanced_options:calibration_iterations': 3,
}
sess = ort.InferenceSession(
    MODEL_PATH,
    providers=['TIDLExecutionProvider'],
    provider_options=[compile_options]
)

input_name = sess.get_inputs()[0].name

calib_images =
os.listdir(CALIBRATION_DATA_DIR)[:compile_options['advanced_options:
calibration_frames']]
for img_name in calib_images:
    img_path = os.path.join(CALIBRATION_DATA_DIR, img_name)

    img = cv2.imread(img_path)
```

```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
img = img.astype(np.float32) / 255.0
img = np.transpose(img, (2, 0, 1))
input_tensor = np.expand_dims(img, axis=0)

_ = sess.run(None, {input_name: input_tensor})
```

Додаток Г

Програмна реалізація конвеєра захоплення, обробки та трансляції відеопотоку

```
import cv2
import time
import numpy as np
import onnxruntime as ort

SERVER_IP = "192.168.1.100"
UDP_PORT = 5000
MODEL_PATH = "tidl_artifacts/yolov8n_quantized.onnx"
CONF_THRESHOLD = 0.5

cap_pipeline = (
    "v4l2src device=/dev/video0 ! "
    "video/x-raw, width=640, height=480, framerate=30/1 ! "
    "videoconvert ! appsink drop=true max-buffers=1"
)

out_pipeline = (
    "appsrc ! videoconvert ! "
    "v4l2h264enc extra-
controls=\"controls,h264_profile=4,video_bitrate=1000000\" ! "
    "rtph264pay config-interval=1 pt=96 ! "
    f"udpsink host={SERVER_IP} port={UDP_PORT} sync=false"
)

cap = cv2.VideoCapture(cap_pipeline, cv2.CAP_GSTREAMER)
out = cv2.VideoWriter(out_pipeline, cv2.CAP_GSTREAMER, 0, 30.0,
(640, 480), True)

providers = ['TIDLExecutionProvider', 'CPUExecutionProvider']
provider_options = [{'tidl_tools_path': '/opt/tidl_tools',
'artifacts_folder': 'tidl_artifacts'}, {}]
session = ort.InferenceSession(MODEL_PATH, providers=providers,
provider_options=provider_options)
```

```
input_name = session.get_inputs()[0].name

prev_time = time.time()

try:
    while True:
        ret, frame = cap.read()
        if not ret:
            time.sleep(1)
            continue

        input_tensor = cv2.resize(frame, (320, 320))
        input_tensor = cv2.cvtColor(input_tensor, cv2.COLOR_BGR2RGB)
        input_tensor = input_tensor.astype(np.float32) / 255.0
        input_tensor = np.transpose(input_tensor, (2, 0, 1))
        input_tensor = np.expand_dims(input_tensor, axis=0)

        outputs = session.run(None, {input_name: input_tensor})
        detections = outputs[0]

        curr_time = time.time()
        fps = 1 / (curr_time - prev_time)
        prev_time = curr_time

        cv2.putText(frame, f"Edge AI FPS: {fps:.1f}", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
        cv2.putText(frame, "Engine: TIDL (C7x/MMA)", (10, 60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 0), 2)

        out.write(frame)

except KeyboardInterrupt:
    pass
finally:
    cap.release()
    out.release()
```

Додаток Д

Програмна реалізація клієнтської підсистеми прийому та декодування відеопотоку

```
import cv2
import sys

UDP_PORT = 5000
receive_pipeline = (
    f"udpsrc port={UDP_PORT} caps=\"application/x-rtp,
media=(string)video, "
    f"clock-rate=(int)90000, encoding-name=(string)H264,
payload=(int)96\" ! "
    "rtph264depay ! h264parse ! avdec_h264 ! videoconvert ! appsink
drop=true sync=false"
)
cap = cv2.VideoCapture(receive_pipeline, cv2.CAP_GSTREAMER)

if not cap.isOpened():
    sys.exit(1)
try:
    while True:
        ret, frame = cap.read()
        if not ret:
            continue
        cv2.imshow("BeagleBone AI-64 Object Tracking System", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

except KeyboardInterrupt:
    pass
finally:
    cap.release()
    cv2.destroyAllWindows()
```