

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ ЗАКЛАД  
«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики  
та інформаційних технологій


Кафедра інформаційних технологій та систем

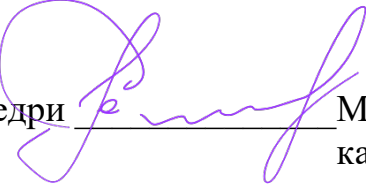
**Руденко Олександр Андрійович**

**Дослідження технологій розробки системи комп'ютерного зору для  
задач навігації автономного транспортного засобу**

**кваліфікаційна робота  
здобувача вищої освіти другого (магістерського) рівня  
освітньої програми «Мультимедійні системи»  
за спеціальністю 121 „Інженерія програмного забезпечення”**

Особистий підпис  Олександр РУДЕНКО

Науковий керівник  Галина КОЗУБ,  
кандидат технічних наук,  
доцент кафедри інформаційних  
технологій та систем

Завідувач кафедри  Микола СЕМЕНОВ,  
кандидат педагогічних наук,  
доцент кафедри інформаційних  
технологій та систем

Лубни – 2026

## АННОТАЦІЯ

**Руденко О.А.**

**Тема:** Дослідження технологій розробки системи комп'ютерного зору для задач навігації автономного транспортного засобу

**Спеціальність:** 121 „Інженерія програмного забезпечення”

**Установа:** ДЗ ЛНУ імені Тараса Шевченка, 2026 р.

**Кваліфікаційна робота містить:** 104 стор., 6 таблиць, 17 рисунків, 37 джерел.

**Об'єкт дослідження** – процеси сприйняття навколишнього середовища та оцінки глибини сцени на основі зображень, отриманих від камер автономного транспортного засобу.

**Предмет дослідження** – технології та методи розробки системи комп'ютерного зору для визначення відстані до об'єктів у задачах навігації автономного транспорту.

**Мета роботи** – розробка та експериментальна оцінка системи виявлення об'єктів і визначення відстані до них на основі сучасних моделей комп'ютерного зору з використанням OpenCV, TensorFlow та датасету KITTI.

**Результати роботи.** Проведено порівняльний аналіз існуючих моделей глибокого навчання для комп'ютерного зору; розроблено систему визначення відстані на основі TensorFlow та OpenCV; здійснено практичну демонстрацію роботи створеної системи.

**Ключові слова:** КОМП'ЮТЕРНИЙ ЗІР, АВТОНОМНИЙ ТРАНСПОРТНИЙ ЗАСІБ, ГЛИБОКЕ НАВЧАННЯ, TENSERFLOW, OPENCV, KITTI DATASET, СИСТЕМА НАВІГАЦІЇ.

## ABSTRACT

**Rudenko O.A.**

**Topic:** Research on technologies for developing a computer vision system for autonomous vehicle navigation tasks

**Specialty:** 121 "Software Engineering"

**Institution:** Taras Shevchenko Luhansk National University, 2025

**Qualification thesis contains:** 104 pages., 6 tables, 17 images, 37 references.

**Research object** – the processes of perception of the surrounding environment and scene depth estimation based on images acquired from cameras of an autonomous vehicle.

**Research subject** – technologies and methods for developing a computer vision system designed to measure distances to objects in autonomous vehicle navigation tasks.

**The purpose of the study** – Development and experimental evaluation of the system for identifying objects and identifying them based on current models of computer surveillance using OpenCV, TensorFlow and the KITTI dataset.

**Research results:** A comparative analysis of existing deep learning models for computer vision was performed; a distance estimation system based on TensorFlow and OpenCV was developed; and practical demonstrations of the developed system's functionality were conducted.

**Keywords:** COMPUTER VISION, AUTONOMOUS VEHICLE, DEEP LEARNING, TENSERFLOW, OPENCV, KITTI DATASET, NAVIGATION SYSTEM.

## ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ НАВІГАЦІЇ АТЗ .....	13
1.1. КОМП'ЮТЕРНИЙ ЗІР ЯК СКЛАДОВА ПРОГРАМНИХ СИСТЕМ АВТОНОМНОЇ НАВІГАЦІЇ .....	13
1.2. ГЕОМЕТРИЧНІ ОСНОВИ ФОРМУВАННЯ ЗОБРАЖЕННЯ ТА МОДЕЛЬ КАМЕРИ ...	18
1.2.1. Пінхол-модель камери .....	19
1.2.2. Матриця внутрішньої калібровки камери .....	19
1.2.3. Калібрування камери та корекція дисторсій .....	20
1.2.4. Значення геометричної моделі для задач навігації АТЗ.....	21
1.2.5. Обмеження геометричних моделей у реальних сценах .....	21
1.3. МЕТОДИ ОЦІНКИ ГЛИБИНИ СЦЕНИ ТА ВІДСТАНЕЙ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ .....	22
1.3.1. СТЕРЕОСКОПІЧНІ МЕТОДИ ОЦІНКИ ГЛИБИНИ .....	22
1.4. ДАТАСЕТИ ТА ЕТАЛОННІ НАБОРИ ДАНИХ ДЛЯ ДОСЛІДЖЕННЯ СИСТЕМ КОМП'ЮТЕРНОГО ЗОРУ В АВТОНОМНИХ ТРАНСПОРТНИХ ЗАСОБАХ.....	25
1.4.1. Загальні вимоги до датасетів для систем навігації АТЗ .....	26
1.4.2. Огляд датасетів для комп'ютерного зору в автономних системах .....	26
1.4.3. Датасет KITTI та його роль у дослідженнях оцінки глибини.....	27
1.4.4. Значення KITTI для експериментальних досліджень.....	27
1.4.5. Висновки до підрозділу.....	28
1.5. КЛАСИФІКАЦІЯ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ОЦІНКИ ГЛИБИНИ СЦЕНИ .....	28
1.5.1. Геометричні методи камерної оцінки глибини .....	28
1.5.2. Монокулярні аналітичні підходи.....	29
1.5.3. Нейромережеві методи монокулярної оцінки глибини.....	29
1.5.4. Порівняльна характеристика підходів .....	30



1.6. ГІБРИДНІ ПІДХОДИ ДО ОЦІНКИ ГЛИБИНИ ТА ВІДСТАНІ В СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ .....	31
Висновки до розділу 1 .....	33
<b>РОЗДІЛ 2 ВИБІР ІНСТРУМЕНТІВ ТА ПРОГРАМНОГО СЕРЕДОВИЩА ДЛЯ РОЗРОБКИ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ .....</b>	<b>35</b>
2.1. ЗАГАЛЬНІ ВИМОГИ ДО ПРОГРАМНОГО СЕРЕДОВИЩА СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ АТЗ .....	35
2.2. ЗАГАЛЬНА СХЕМА ІНСТРУМЕНТІВ І ПРОГРАМНОГО СЕРЕДОВИЩА СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ .....	36
2.2.1. <i>Загальна архітектура інструментів</i> .....	36
2.2.2. <i>Вибір програмних інструментів та середовища реалізації</i> .....	38
2.3. ВИКОРИСТАННЯ БІБЛІОТЕКИ OPENCV У ЗАДАЧАХ КОМП'ЮТЕРНОГО ЗОРУ .	38
2.4. ВИКОРИСТАННЯ ПЛАТФОРМИ TENSORFLOW ТА МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ В СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ .....	40
2.4.1. <i>Загальна огляд платформи TenslorFlow</i> .....	40
2.4.2. <i>Монокулярна оцінка глибини (Monocular Depth Estimation)</i> .....	42
2.4.3. <i>Обмеження та джерела похибок монокулярної оцінки глибини в реальних сценах</i> .....	44
2.4.4. <i>Ф'южн та згладжування оцінок глибини</i> .....	46
2.5. ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ОЦІНКИ ГЛИБИНИ, ВИКОРИСТАНИХ У ДОСЛІДЖЕННІ .....	48
Висновки до розділу 2 .....	50
<b>РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ВИМІРЮВАННЯ ВІДСТАНІ ДО ОБ'ЄКТІВ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ АТЗ .....</b>	<b>52</b>
3.1. МЕТА ТА ПОСТАНОВКА ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ.....	52
3.2. АРХІТЕКТУРА ТА СТРУКТУРА ПРОГРАМНОГО ПРОЄКТУ .....	55
3.3. МЕТОДИКА ВИМІРЮВАННЯ ВІДСТАНІ ДО ОБ'ЄКТІВ .....	57
3.4. ВІЗУАЛЬНІ РЕЗУЛЬТАТИ ВИМІРЮВАННЯ ВІДСТАНІ ДО ОБ'ЄКТІВ.....	60

3.4.1. Монокулярна геометрична оцінка відстані (pinhole) .....	61
3.4.2. Стереоскопічна оцінка відстані на основі датасету KITTI .....	63
3.4.3. Монокулярна нейромережева оцінка глибини (MDE) .....	64
3.4.4. Вимірювання відстані за калібрувальними маркерами ArUco .....	65
3.5. ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ВИМІРЮВАННЯ ВІДСТАНІ НА ОДНАКОВИХ СЦЕНАХ .....	66
3.5.1. Повний аналіз всіх наявних сцен і об'єктів .....	66
3.5.2. Аналіз результатів у складних умовах освітлення .....	69
3.5.2. Аналіз ефективності ф'южну методів вимірювання відстані.....	70
3.6. АНАЛІЗ ПРОДУКТИВНОСТІ ТА ЧАСОВИХ ХАРАКТЕРИСТИК ПІДХОДІВ .....	72
Висновки до розділу 3 .....	74
ВИСНОВКИ .....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	79
ДОДАТОК А СЕРТИФІКАТ АПРОБАЦІЇ І ВПРОВАДЖЕННЯ.....	84
ДОДАТОК Б ВИХІДНИЙ КОД ДОДАТКУ .....	85

## ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AT3	–	автоматизований транспортний засіб
CV	–	Computer Vision (комп'ютерний зір)
DL	–	Deep Learning (глибоке навчання)
CNN	–	Convolutional Neural Network (згортова нейронна мережа)
MDE	–	Monocular Depth Estimation (монокулярна оцінка глибини)
ROI	–	Region of Interest (область інтересу)
BBox	–	Bounding Box (обмежувальна рамка об'єкта)
OpenCV	–	Open Source Computer Vision Library
KITTI	–	Karlsruhe Institute of Technology and Toyota Technological Institute
GT	–	Ground Truth (еталонні, істинні дані)
FPS	–	Frames Per Second (кількість кадрів за секунду)
RMSE	–	Root Mean Square Error (середньоквадратична помилка)
AbsRel	–	Absolute Relative Error (середня відносна похибка)
PnP	–	Perspective-n-Point (визначення пози об'єкта відносно камери)
ArUco	–	бібліотека маркерів для оцінки пози та орієнтації
SGBM	–	Semi-Global Block Matching (напівглобальне блочне зіставлення)
BM	–	Block Matching (блочне зіставлення)
EMA	–	Exponential Moving Average (експоненційне згладжування)
ADAS	–	Advanced Driver Assistance Systems (системи допомоги водієві)

## ВСТУП

**Актуальність теми дослідження.** Розвиток автономних транспортних засобів є одним із пріоритетних напрямів сучасних інформаційних технологій, що поєднує досягнення комп'ютерного зору, штучного інтелекту та програмної інженерії. Ключовим компонентом таких систем є програмне забезпечення комп'ютерного зору, яке забезпечує сприйняття навколишнього середовища, зокрема виявлення об'єктів та визначення відстані до них, що безпосередньо впливає на безпеку та ефективність навігації автономного транспорту.

Процес створення програмного забезпечення для систем комп'ютерного зору автономних транспортних засобів характеризується високою складністю та потребує поєднання сучасних алгоритмів обробки зображень із методами глибокого навчання. Особливої уваги потребує задача визначення відстані до об'єктів за даними відеокамер, оскільки вона має критичне значення для прийняття навігаційних рішень у реальному часі.

Сучасні моделі глибокого навчання, зокрема YOLOv8, YOLOv7, EfficientDet та Vision Transformers, демонструють високі показники точності та швидкодії у задачах детекції об'єктів. Водночас ефективність їх застосування для оцінки відстані в системах автономної навігації потребує детального аналізу та експериментального порівняння. Використання фреймворків TensorFlow та бібліотеки OpenCV створює технологічну основу для розробки, тестування та оптимізації таких систем.

У зв'язку з цим дана дипломна робота присвячена дослідженню технологій і методів розробки системи комп'ютерного зору для визначення відстані до об'єктів у задачах навігації автономного транспорту, а також створенню та експериментальній перевірці відповідної програмної реалізації є актуальною.

**Об'єкт дослідження** – процес створення програмного забезпечення для систем комп'ютерного зору автономних транспортних засобів.

**Предмет дослідження** – технології та методи розробки системи комп’ютерного зору для визначення відстані до об’єктів у задачах навігації автономного транспорту.

**Мета роботи** – дослідити існуючі технології комп’ютерного зору, порівняти сучасні моделі глибокого навчання (YOLOv8, YOLOv7, EfficientDet, Vision Transformers); визначити найбільш ефективні інструменти та алгоритми для реалізації системи виявлення та оцінки відстані до об’єктів; створити й протестувати програмну реалізацію системи на базі бібліотеки OpenCV і моделі TensorFlow з використанням датасету KITTI; оцінити ефективність розробленої системи за показниками точності, швидкості та надійності.

Досягнення цієї мети вимагає розв’язання таких завдань:

- Проаналізувати сучасні тенденції розвитку технологій комп’ютерного зору у сфері автономних транспортних засобів.
- Здійснити порівняльний аналіз ефективності сучасних моделей глибокого навчання, таких як TensorFlow, YOLOv8, YOLOv7, EfficientDet, Vision Transformers.
- Визначити оптимальну архітектуру системи для ефективної оцінки відстаней на основі аналізу зображень.
- Реалізувати та протестувати систему визначення відстані з використанням бібліотеки OpenCV та моделі TensorFlow на базі відкритого датасету KITTI.
- Оцінити ефективність системи за основними показниками: точністю, швидкістю та стабільністю роботи.

**Методи дослідження.** Використовуються теоретичні методи, що включають аналіз, порівняння та систематизацію науково-технічних джерел, а також емпіричні методи – практичні експерименти з навчання та тестування нейронних мереж на реальних даних.

**Наукова новизна** дослідження полягає у поєднанні класичних геометричних методів комп'ютерного зору (pinhole, stereo disparity, PnP) з нейромережевими моделями глибини (MiDaS) у єдиній системі оцінки відстані; у виконанні об'єктно-орієнтованої оцінки глибини з використанням детекції об'єктів та статистичної обробки карт глибини в межах областей інтересу; відтворення експериментальної оцінки точності та продуктивності на стандартному датасеті KITTI.

**Практичне значення** дослідження. Результати дослідження можуть бути використані у реальних системах автономного керування транспортними засобами, а також у навчальному процесі під час викладання курсів, пов'язаних з розробкою та експлуатацією систем комп'ютерного зору.

**Особистий внесок** автора полягає у проектуванні та програмній реалізації системи комп'ютерного зору для оцінки відстані до об'єктів із використанням бібліотеки OpenCV та платформи TensorFlow. Реалізації бінокулярних та монокулярних методів оцінки відстані, інтегрована нейромережових моделей детекції об'єктів, а також експериментальної перевірки та аналізу отриманих результатів.

Вірогідність отриманих результатів і сформульованих висновків забезпечується науково обґрунтованою методологією дослідження, відповідністю використаних методів поставленим меті та завданням роботи, комплексним кількісним і якісним аналізом теоретичних та емпіричних матеріалів.

**Апробація і впровадження результатів магістерської роботи** здійснювалися шляхом доповіді основних положень і результатів дослідження на **IV Міжнародній науково-практичній конференції**, що відбулася у червні 2025 року в місті Дрезден (Німеччина), за підсумками якої опубліковано наукову статтю за темою « ЗАСТОСУВАННЯ МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ В ПРОГРАМНИХ РІШЕННЯХ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ АВТОНОМНИХ СИСТЕМ » [14].

**Структура магістерської роботи** складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 104 сторінки, з яких 66 сторінок припадає на основний зміст дослідження, 6 таблиць та 17 рисунків. Список використаних джерел налічує 37 найменування.

**У першому розділі** розглянуто теоретичні та методологічні основи побудови програмних систем комп'ютерного зору для задач вимірювання відстані до об'єктів. Проаналізовано роль комп'ютерного зору в системах навігації автономних транспортних засобів, геометричні моделі камери, класичні методи оцінки просторових параметрів сцени, а також сучасні нейромережеві підходи до детекції об'єктів і відновлення глибини. Окрему увагу приділено інженерним аспектам програмної реалізації та архітектурним підходам до побудови модульних систем комп'ютерного зору.

**У другому розділі** обґрунтовано вибір програмних інструментів, бібліотек і середовищ розробки для реалізації досліджуваної системи. Розглянуто використання бібліотеки OpenCV для низько- та середньорівневої обробки зображень, платформи TensorFlow для реалізації нейромережевих моделей, а також датасету KITTI як еталонного набору даних для експериментальної оцінки. Описано програмну архітектуру системи, модульний принцип побудови, структуру проєкту та методи інтеграції окремих компонентів у єдиний програмний конвеєр.

**У третьому розділі** наведено результати експериментальних досліджень методів вимірювання відстані до об'єктів. Описано методику експериментів, реалізацію бінокулярного стереоскопічного підходу, монокулярних методів на основі геометрії pinhole, нейромережевої оцінки глибини та калібрувальних маркерів ArUco. Представлено результати порівняльного аналізу точності, досліджено вплив умов освітлення та дистанції до об'єктів, розглянуто стратегії ф'южну та згладжування оцінок. Також виконано аналіз продуктивності розробленої системи та визначено обмеження її застосування в режимі реального часу.

**У висновках** узагальнено основні результати роботи, сформульовано практичні рекомендації щодо використання розроблених програмних рішень і окреслено напрями подальших досліджень. У додатках наведено фрагменти програмного коду та допоміжні матеріали, що забезпечують відтворюваність експериментів.

**У додатках** представлено сертифікат впровадження, фрагменти програмного коду розробленої системи, допоміжні скрипти для обробки даних та проведення експериментів, а також матеріали, що забезпечують відтворюваність отриманих результатів.



## **РОЗДІЛ 1**

### **АНАЛІЗ СУЧАСНИХ МЕТОДІВ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ НАВІГАЦІЇ АТЗ**

#### **1.1. Комп'ютерний зір як складова програмних систем автономної навігації**

Сучасний розвиток автономних транспортних засобів (АТЗ) нерозривно пов'язаний із застосуванням методів комп'ютерного зору (Computer Vision, CV), які забезпечують сприйняття навколишнього середовища, інтерпретацію сцен дорожнього руху та прийняття навігаційних рішень у реальному часі. На відміну від традиційних інформаційних систем, системи комп'ютерного зору працюють безпосередньо з візуальними даними, що дозволяє моделювати процес людського зорового сприйняття та використовувати його для аналізу складних динамічних сцен. У наукових дослідженнях з інженерії програмного забезпечення підкреслюється, що ефективність таких систем значною мірою залежить від правильного вибору архітектурних підходів, алгоритмічних рішень та засобів програмної реалізації [1], [2].

Комп'ютерний зір є ключовим компонентом програмного забезпечення автономних систем, оскільки саме візуальна інформація містить найбільшу кількість семантичних ознак: положення об'єктів, їх форму, розміри, взаємне розташування та динаміку. У роботах, присвячених аналізу та обробці зображень у прикладних інформаційних системах, відзначається універсальність методів комп'ютерного зору та можливість їх адаптації до різних предметних областей, зокрема задач безпеки, навігації та моніторингу [4], [5]. Водночас у наукових публікаціях з комп'ютерного зору та програмної інженерії зазначається, що практична ефективність автономних систем визначається не лише точністю окремих алгоритмів, а й архітектурною організацією програмного забезпечення, здатністю до оптимального використання обчислювальних ресурсів та можливістю масштабування під реальні умови експлуатації [4], [6].

Особливого значення набувають питання програмної архітектури, організації обчислювальних пайплайнів та інтеграції алгоритмів комп'ютерного зору в складні програмні системи автономного керування. У сучасних дослідженнях наголошується, що саме інженерні рішення щодо структури програмного забезпечення, розподілу обчислень та взаємодії між модулями визначають стабільність і відтворюваність результатів роботи систем комп'ютерного зору [1], [2], [3].

У контексті автономної навігації комп'ютерний зір виконує низку критично важливих функцій, серед яких: детекція та класифікація об'єктів дорожньої інфраструктури, оцінка відстаней до перешкод, аналіз траєкторій руху інших учасників дорожнього руху, а також побудова просторової моделі сцени. Загальні принципи побудови таких програмних систем, а також роль алгоритмічної складової у забезпеченні точності та надійності обробки візуальних даних, детально розглядаються в дослідженнях з сучасних інформаційних технологій та цифрових систем [3], [6]. Дослідження застосування методів глибокого навчання в програмних рішеннях комп'ютерного зору демонструють, що поєднання класичних алгоритмів обробки зображень із неймережевими моделями дозволяє досягти високої точності та стабільності роботи систем у змінних умовах освітлення та середовища [7].

Автономні транспортні засоби використовують комплекс сенсорів, до складу якого можуть входити камери, лідари, радары та інерціальні вимірювальні системи. Проте саме камери залишаються найбільш доступним та універсальним джерелом інформації, оскільки вони забезпечують високу роздільну здатність, багатство візуальних ознак та відносно низьку вартість. У наукових публікаціях з комп'ютерного зору та програмної інженерії наголошується, що камерні системи є основою для побудови більшості сучасних алгоритмів аналізу зображень, включаючи задачі детекції, сегментації та оцінки глибини [4], [7]. У зв'язку з цим значна частина сучасних

досліджень зосереджена на розробці камерно-орієнтованих систем комп'ютерного зору для автономного керування [14], [15].

Окрему увагу в наукових роботах приділено питанням програмної реалізації та оптимізації обробки зображень. Застосування GPU-обчислень, паралельних алгоритмів та спеціалізованих бібліотек дозволяє значно підвищити продуктивність систем комп'ютерного зору, що є критично важливим для обробки даних у реальному часі [6]. Використання сучасних програмних платформ і відкритих бібліотек також розглядається як ключовий чинник забезпечення масштабованості, підтримуваності та повторного використання програмних компонентів [1], [6]. Зокрема, широке застосування бібліотек OpenCV та TensorFlow забезпечує відтворюваність експериментів і спрощує інтеграцію алгоритмів комп'ютерного зору у прикладні програмні системи.

У сучасних автономних транспортних засобах системи комп'ютерного зору реалізуються у вигляді багаторівневого ієрархічної структури обробки візуальних сигналів, який охоплює повний цикл обробки візуальної інформації - від отримання сирих даних сенсорів до формування високорівневих навігаційних рішень. Такий пайплайн охоплює послідовні етапи захоплення зображень, їх попередньої обробки, виділення та інтерпретації ознак, оцінки просторових параметрів сцени та передачі узагальнених результатів до модулів планування руху автономного транспортного засобу.

На початковому рівні обробки візуальної інформації здійснюється зчитування зображень з камер та їх базова обробка, що може включати корекцію оптичних спотворень, нормалізацію яскравості, фільтрацію шумів і синхронізацію потоків даних. Ці операції мають важливе значення для забезпечення стабільності подальших етапів аналізу, особливо в умовах змінного освітлення та складних погодних умов. Дослідження у сфері програмної обробки графічних даних підкреслюють, що оптимізація низькорівневих обчислень та ефективне використання апаратних ресурсів

істотно впливають на загальну продуктивність системи комп'ютерного зору [6].

Наступним етапом багаторівневої системи обробки та інтерпретації візуальних даних є виділення та інтерпретація об'єктів сцени. На цьому рівні використовуються алгоритми детекції та класифікації об'єктів, які дозволяють ідентифікувати транспортні засоби, пішоходів, дорожні знаки, розмітку та інші елементи інфраструктури. Сучасні підходи ґрунтуються переважно на методах глибокого навчання, що забезпечують високу точність навіть у складних динамічних сценах. Водночас ефективність таких рішень значною мірою залежить від програмної реалізації та здатності системи обробляти великі обсяги даних у реальному часі [7], [15].

У загальній архітектурі автономного транспортного засобу комп'ютерний зір формує рівень візуального сприйняття та аналізу середовища, який відповідає за перетворення сирих сенсорних даних у структуроване подання навколишньої сцени. На цьому рівні відбувається перехід від фізичних сигналів до семантично значущих об'єктів і станів, таких як «перешкода», «транспортний засіб», «пішохід» або «вільна зона руху». У працях, присвячених архітектурі складних програмних систем, підкреслюється, що коректна реалізація цього рівня є визначальним чинником ефективності всієї системи автономного керування [1], [3].

Комп'ютерний зір у цьому контексті виступає не ізольованим набором алгоритмів, а програмно-алгоритмічною підсистемою, інтегрованою з іншими модулями АТЗ. Результати візуального аналізу передаються до модулів трекінгу, прогнозування та планування руху, де вони використовуються для прийняття рішень у реальному часі. Така взаємодія вимагає чіткої формалізації інтерфейсів та синхронізації обчислень, що підкреслюється у сучасних дослідженнях з інженерії програмного забезпечення [2], [6].

Ключовим завданням комп'ютерного зору в автономних системах є не лише розпізнавання об'єктів, але й оцінка їх просторового положення відносно транспортного засобу. Саме на цьому етапі відбувається перехід від

двовимірному аналізу зображень до тривимірного представлення сцени. Визначення відстаней до об'єктів, їх відносної глибини та просторової структури середовища є критично важливим для безпечної навігації, оскільки безпосередньо впливає на рішення щодо швидкості руху, гальмування та маневрування.

Реалізація таких функцій потребує чіткого програмного розділення відповідальностей між компонентами системи. Архітектура CV-системи автономного транспортного засобу зазвичай передбачає модульну організацію, у якій окремі компоненти відповідають за детекцію об'єктів, оцінку глибини сцени, трекінг та агрегацію результатів. Подібний підхід дозволяє масштабувати систему, замінювати або вдосконалювати окремі алгоритми без порушення цілісності програмного рішення, що узгоджується з сучасними підходами до розробки складних програмних систем [1], [4].

Важливим аспектом є також інтеграція систем комп'ютерного зору з іншими програмними підсистемами автономного транспортного засобу. Результати візуального аналізу сцени передаються до модулів планування траєкторії та керування, де вони поєднуються з даними інших сенсорів. Така взаємодія вимагає чітко визначених інтерфейсів, часової узгодженості та гарантованої надійності програмних компонентів. Дослідження у сфері розробки складних інформаційних систем підкреслюють, що саме програмна архітектура відіграє визначальну роль у забезпеченні стабільності та відтворюваності результатів [2], [3].

У зв'язку з цим оцінка глибини сцени та відстаней до об'єктів не може розглядатися ізольовано від програмної реалізації. Алгоритмічні рішення мають бути адаптовані до обмежень обчислювальних ресурсів, вимог реального часу та умов експлуатації автономного транспортного засобу. Саме тому в сучасних дослідженнях спостерігається тенденція до поєднання класичних геометричних методів, що мають чітке фізичне обґрунтування, з нейромережевими підходами, здатними узагальнювати складні закономірності у візуальних даних [7], [8].

Особливістю систем комп'ютерного зору для АТЗ є жорсткі вимоги до затримок обробки даних. Навігаційні рішення повинні формуватися у реальному або квазі-реальному часі, що обмежує складність використовуваних алгоритмів і накладає вимоги на їх оптимізацію. У цьому контексті особливого значення набуває використання паралельних обчислень та апаратного прискорення, зокрема із застосуванням GPU [6].

Дослідження показують, що навіть високоточні алгоритми втрачають практичну цінність, якщо їх час виконання перевищує допустимі межі для систем автономного керування [14], [15]. Тому в сучасних програмних рішеннях для комп'ютерного зору активно використовуються компромісні підходи, які поєднують прийнятну точність з обмеженими обчислювальними витратами.

Таким чином, комп'ютерний зір у системах навігації автономних транспортних засобів слід розглядати як цілісну програмно-алгоритмічну підсистему, що інтегрує методи обробки зображень, машинного навчання та інженерні підходи до побудови програмної архітектури. Для подальшого аналізу таких систем доцільно детально розглянути математичні моделі формування зображення та геометричні принципи проєкції тривимірної сцени на площину зображення, які є основою для оцінки просторових параметрів та відстаней до об'єктів і становлять предмет наступного підрозділу.

## **1.2. Геометричні основи формування зображення та модель камери**

Ефективне функціонування систем комп'ютерного зору (CV) у задачах навігації автономних транспортних засобів (АТЗ) базується на коректному математичному описі процесу формування зображення. Незалежно від складності подальших алгоритмів аналізу сцени, базовим елементом візуально-аналітичного контуру є камера, яка здійснює проєкцію тривимірного простору на двовимірну площину зображення. У зв'язку з цим геометричні моделі камери відіграють визначальну роль у задачах оцінки глибини сцени, визначення відстаней до об'єктів та побудови просторового подання навколишнього середовища.

У більшості сучасних CV-систем для АТЗ використовується пінхол-модель камери (pinhole camera model), яка є спрощеним, але достатньо точним наближенням реального процесу формування зображення. В межах цієї моделі камера розглядається як ідеальний оптичний пристрій, у якому всі промені світла проходять через одну точку - центр проєкції, а зображення формується на площині матриці. Такий підхід дозволяє отримати чіткий аналітичний зв'язок між координатами об'єкта у тривимірному просторі та його проєкцією на зображенні.

### *1.2.1. Пінхол-модель камери*

Математично процес проєкції у пінхол-моделі описується співвідношенням між просторовими координатами точки  $(X, Y, Z)$  та її координатами на зображенні  $(x, y)$ :

$$x = \frac{f * X}{Z}, \quad y = \frac{f * Y}{Z}$$

де  $f$  - фокусна відстань камери,  $Z$  - глибина точки відносно оптичного центру. Таким чином, координати точки на зображенні залежать від її просторового положення та параметрів камери, що безпосередньо пов'язує задачу оцінки відстані з точністю геометричної моделі.

Для переходу до піксельних координат у цифровому зображенні використовується розширена форма пінхол-моделі, яка враховує параметри внутрішньої калібровки камери.

### *1.2.2. Матриця внутрішньої калібровки камери*

Внутрішні параметри камери описуються матрицею калібрування  $K$ , яка має вигляд:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

де  $f_x$  та  $f_y$  - фокусні відстані вздовж осей зображення, а  $c_x$  та  $c_y$  - координати головної точки. Методики визначення цих параметрів докладно

описані в класичних роботах з калібрування камер [26] та широко застосовуються у сучасних системах комп'ютерного зору.

Окрім внутрішніх параметрів, важливу роль відіграють зовнішні параметри камери, які описують її положення та орієнтацію у просторі відносно світової системи координат. Разом внутрішні та зовнішні параметри формують повну модель проєкції, що використовується у задачах тривимірної реконструкції та оцінки глибини сцени.

### *1.2.3. Калібрування камери та корекція дисторсій*

Коректність геометричних методів комп'ютерного зору безпосередньо залежить від точності калібрування камерної системи. Калібрування дозволяє встановити відповідність між координатами пікселів зображення та напрямками променів у просторі, що є необхідною умовою для будь-яких подальших просторових обчислень. У класичних роботах з калібрування камер наголошується, що навіть незначні похибки у визначенні внутрішніх параметрів можуть призводити до суттєвих спотворень оцінки відстаней [26].

У практичних системах автономної навігації калібрування камер виконується на етапі підготовки апаратного комплексу, однак у процесі експлуатації можливі додаткові джерела похибок, пов'язані з вібраціями, температурними змінами або механічними зсувами. Це зумовлює необхідність врахування похибок калібрування при інтерпретації результатів оцінки глибини.

У реальних системах формування зображення ускладнюється наявністю оптичних спотворень, зумовлених властивостями об'єктива. Найбільш поширеними є радіальні та тангенціальні дисторсії, які призводять до викривлення прямих ліній на зображенні. Для компенсації цих ефектів застосовуються процедури калібрування та корекції дисторсій, реалізовані, зокрема, у бібліотеці OpenCV [25].

Математичні основи формування зображення є фундаментом для подальшого розвитку алгоритмів оцінки глибини сцени. Саме на цих принципах будуються як класичні геометричні методи, зокрема бінокулярна



стерео-обробка, так і сучасні нейромережеві підходи, які, хоча й використовують навчання на даних, все одно опосередковано спираються на фізичні закономірності проєкції [8–12].

#### *1.2.4. Значення геометричної моделі для задач навігації АТЗ*

Геометрична модель камери є фундаментом для побудови більш складних алгоритмів комп'ютерного зору, що використовуються в системах навігації автономних транспортних засобів. Саме на її основі реалізуються методи стереозору, бінокулярної оцінки глибини, монокулярної оцінки відстані та нейромережеві підходи до реконструкції просторової структури сцени.

У контексті АТЗ пінхол-модель та матриця калібрування дозволяють здійснювати перехід від двовимірного аналізу зображень до тривимірного опису навколишнього середовища, що є необхідною умовою для прийняття навігаційних рішень. Зокрема, оцінка відстаней до об'єктів, аналіз безпечних зон руху та прогнозування траєкторій інших учасників дорожнього руху безпосередньо залежать від точності геометричної моделі камери.

Таким чином, пінхол-модель камери та процедура її калібрування формують теоретичну основу для подальшого розгляду методів оцінки глибини сцени. У наступному підрозділі буде детально розглянуто основні підходи до визначення глибини та відстаней у системах комп'ютерного зору, зокрема стереоскопічні та монокулярні методи, що активно застосовуються в автономних транспортних засобах.

#### *1.2.5. Обмеження геометричних моделей у реальних сценах*

Геометричні моделі формування зображення базуються на низці ідеалізованих припущень, зокрема щодо точності проєкції та відсутності нелінійних спотворень. У реальних дорожніх сценах ці припущення не завжди виконуються, що призводить до накопичення похибок. Зокрема, складні умови освітлення, відблиски, напівпрозорі поверхні та низька текстурованість можуть знижувати ефективність геометричних методів [24], [32]. Алгоритм

вимірювання відстані показано на рисунку 1.1 і залежить від реального розміру об'єкта ( $h_{real}$ ), а також наскільки точно об'єкт попадає на кадр.

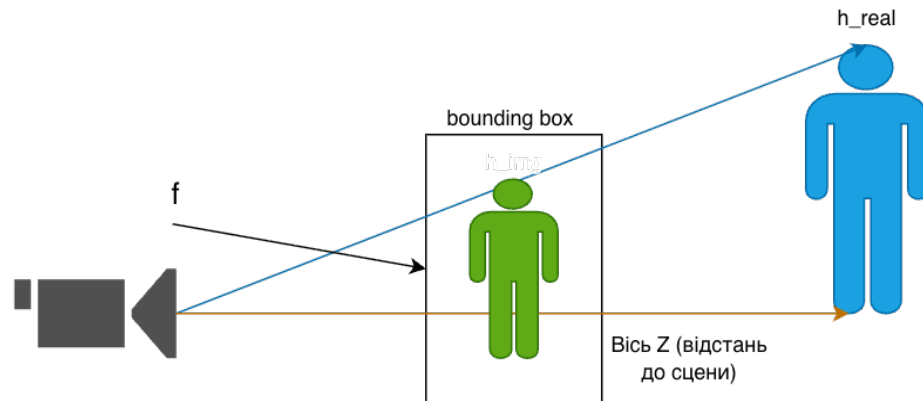


Рисунок 1.1. Геометрична оцінка відстані

Ці обмеження є одним із чинників, які стимулювали розвиток альтернативних підходів до оцінки глибини, зокрема нейромережових методів монокулярної оцінки.

### 1.3. Методи оцінки глибини сцени та відстаней у системах комп'ютерного зору

Оцінка глибини сцени та відстаней до об'єктів є однією з ключових задач комп'ютерного зору (CV) у системах навігації автономних транспортних засобів (АТЗ). Саме здатність коректно визначати просторове положення об'єктів у навколишньому середовищі забезпечує безпечне планування руху, уникнення зіткнень та прийняття адекватних навігаційних рішень у реальному часі.

На відміну від задач детекції та класифікації, які переважно оперують двовимірними ознаками зображення, оцінка глибини потребує відновлення тривимірної структури сцени. Для цього в сучасних CV-системах застосовуються різні підходи, які умовно можна поділити на геометричні методи, монокулярні методи на основі апріорних припущень та нейромережові методи глибинної оцінки

#### 1.3.1. Стереоскопічні методи оцінки глибини

Стереоскопічні (бінокулярні) методи базуються на використанні двох або більше камер, розташованих на відомій відстані одна від одної. Основна

ідея полягає у визначенні диспаратії - зсуву відповідних точок між лівим і правим зображенням стереопари. На основі цього зсуву відновлюється глибина сцени.

Після виконання калібрування та ректифікації камер, що забезпечує горизонтальне вирівнювання епіполярних ліній, глибина  $Z$  для кожної точки може бути обчислена за формулою:

$$Z = \frac{f_x * B}{d},$$

де  $f_x$  - фокусна відстань камери в пікселях,  $B$  - базис між камерами,  $B$  - значення диспаратії. Таким чином, точність оцінки глибини безпосередньо залежить від точності калібрування, якості обчислення диспаратії та стабільності візуальних ознак сцени.

Стереометоди забезпечують метричну оцінку глибини та добре інтерпретуються з фізичної точки зору, що робить їх привабливими для застосування в АТЗ. Проте їх ефективність знижується в умовах низької текстурованості, сильних відблисків, туману або при значних відстанях до об'єктів, де диспаратія стає малою.

### 1.3.2. Монокулярні геометричні підходи

Монокулярні методи оцінки відстаней використовують лише одне зображення та базуються на геометричних припущеннях щодо сцени або відомих фізичних розмірів об'єктів. У таких підходах пінхол-модель камери, розглянута в підрозділі 1.2, поєднується з інформацією про реальні габарити об'єктів певного класу.

Наприклад, відстань до об'єкта може бути оцінена за формулою:

$$Z = \frac{f * H}{h},$$

де  $f$  - фокусна відстань камери,  $H$  - відома реальна висота об'єкта,  $h$  - його висота в пікселях на зображенні. Подібні методи часто застосовуються для оцінки відстані до пішоходів, транспортних засобів або дорожніх знаків.

Перевагою монокулярних геометричних підходів є їхня простота та низькі обчислювальні витрати. Водночас такі методи є чутливими до помилок класифікації об'єктів, змін їхніх реальних розмірів та перспективних спотворень, що обмежує їх застосування в складних реальних сценаріях.

### **1.3.3. Нейромережеві методи монокулярної оцінки глибини (MDE)**

Останніми роками значного поширення набули нейромережеві підходи до монокулярної оцінки глибини (Monocular Depth Estimation, MDE), які дозволяють відновлювати структуру сцени з одного зображення без явних геометричних припущень. Такі методи ґрунтуються на використанні згорткових нейронних мереж або трансформерних архітектур, навчених на великих датасетах із глибинною розміткою.

Нейромережеві MDE-моделі здатні відновлювати відносну глибину сцени, використовуючи контекстні та семантичні ознаки, такі як форма об'єктів, перспективні співвідношення та глобальна структура зображення. Це дозволяє їм демонструвати стабільні результати навіть у випадках, коли класичні стереометоди працюють нестабільно.

Основним обмеженням MDE-підходів є відсутність абсолютного метричного масштабу. Отримана карта глибини, як правило, є масштабно-неоднозначною, що вимагає додаткових процедур вирівнювання масштабу або поєднання з геометричними методами для використання в навігаційних задачах АТЗ.

### **1.3.4. Порівняльний аналіз підходів до оцінки глибини**

Кожен з розглянутих підходів має свої переваги та обмеження. Стереоскопічні методи забезпечують фізично обґрунтовану метричну оцінку глибини, але є чутливими до умов зйомки та якості текстур. Монокулярні геометричні підходи прості в реалізації, проте залежать від коректності припущень щодо сцени. Нейромережеві MDE-методи демонструють високу узагальнювальну здатність, але потребують додаткових механізмів масштабування для використання в практичних навігаційних системах.

У сучасних CV-системах АТЗ спостерігається тенденція до комбінування різних методів, що дозволяє компенсувати недоліки окремих підходів. Поєднання геометричних та нейромережевих рішень розглядається як перспективний напрям розвитку систем оцінки глибини сцени.

### **1.3.5. Роль методів оцінки глибини у навігації АТЗ**

Оцінка глибини сцени є одним із ключових компонентів системи візуального аналізу автономного транспортного засобу, оскільки безпосередньо впливає на прийняття навігаційних рішень. Результати роботи алгоритмів стереозору, монокулярних та нейромережевих методів використовуються для формування просторової моделі середовища, оцінки безпечних зон руху та прийняття рішень щодо керування транспортним засобом.

Таким чином, вибір та коректна інтеграція методів оцінки глибини визначають надійність і безпеку всієї системи навігації АТЗ. У подальших розділах буде розглянуто практичну реалізацію зазначених підходів, а також експериментальне дослідження їх ефективності з використанням еталонних датасетів.

## **1.4. Датасети та еталонні набори даних для дослідження систем комп'ютерного зору в автономних транспортних засобах**

Розробка, навчання та об'єктивне порівняння алгоритмів комп'ютерного зору (CV) для систем навігації автономних транспортних засобів (АТЗ) неможливі без використання стандартизованих датасетів. Саме еталонні набори даних забезпечують відтворюваність експериментів, можливість кількісної оцінки якості алгоритмів та коректне порівняння різних підходів у єдиних умовах.

У задачах детекції об'єктів, оцінки глибини сцени та просторового сприйняття середовища датасети відіграють роль не лише джерела навчальних прикладів, але й інструмента валідації ефективності програмних рішень. Для систем АТЗ особливо важливими є набори даних, що відображають реальні

дорожні сцени з різноманітними умовами освітлення, погодними факторами, типами об'єктів та конфігураціями руху.

#### *1.4.1. Загальні вимоги до датасетів для систем навігації АТЗ*

Датасети, орієнтовані на задачі автономної навігації, повинні відповідати низці ключових вимог. По-перше, вони мають містити реалістичні сцени дорожнього руху, отримані в реальних умовах експлуатації транспортних засобів. По-друге, важливою є наявність точної просторової розмітки, яка дозволяє оцінювати не лише семантичні, але й геометричні характеристики сцени.

Для задач оцінки глибини сцени та відстаней особливу роль відіграє наявність:

- калібрувальних параметрів камер;
- стереопар з відомим базисом;
- еталонних карт диспаратії або глибини;
- синхронізованих даних від кількох сенсорів.

Крім того, з позицій інженерії програмного забезпечення важливими є обсяг датасету, структурованість даних та можливість автоматизованої обробки в рамках експериментальних пайплайнів.

#### *1.4.2. Огляд датасетів для комп'ютерного зору в автономних системах*

У сучасних дослідженнях широко використовуються такі датасети, як Cityscapes, Waymo Open Dataset, nuScenes, ApolloScape та KITTI. Кожен з них має власну спеціалізацію та орієнтований на певні аспекти задач автономного керування.

Наприклад, Cityscapes зосереджений на семантичній сегментації міських сцен, nuScenes та Waymo надають багатосенсорні дані з радарів, лідарів і камер, тоді як ApolloScape пропонує детальну розмітку для задач локалізації та картографування. Проте не всі з перелічених датасетів однаково зручні для дослідження задач оцінки глибини сцени з використанням як геометричних, так і нейромережевих підходів.

### *1.4.3. Датасет KITTI та його роль у дослідженнях оцінки глибини*

Одним з найбільш поширених і визнаних еталонних наборів даних у сфері CV для АТЗ є датасет KITTI, розроблений у співпраці Karlsruhe Institute of Technology та Toyota Technological Institute. Датасет KITTI широко використовується в наукових дослідженнях і промислових розробках для оцінки алгоритмів стереозору, монокулярної глибини, детекції об'єктів та трекінгу.

Ключовою перевагою KITTI є наявність:

- каліброваних стереокамер;
- точно відомих параметрів камерної системи;
- еталонних карт диспаратії та глибини;
- розмітки об'єктів дорожньої сцени;
- реальних записів з дорожніх умов.

Завдяки цьому KITTI дозволяє будувати повний пайплайн оцінки глибини: від ректифікації стереопар і обчислення диспаратії до відновлення метричної глибини сцени та оцінки відстаней до об'єктів. Водночас той самий набір даних може використовуватися для тестування нейромережових монокулярних методів (MDE), що створює умови для коректного порівняння різних підходів у єдиному експериментальному середовищі.

### *1.4.4. Значення KITTI для експериментальних досліджень*

Використання KITTI як базового датасету в дослідженнях систем комп'ютерного зору для АТЗ забезпечує низку переваг. По-перше, наявність еталонної глибини дозволяє кількісно оцінювати точність алгоритмів за стандартизованими метриками. По-друге, реалістичність сцен сприяє перевірці алгоритмів у умовах, наближених до реальної експлуатації.

Крім того, KITTI підтримує аналіз обмежень і проблемних сценаріїв, таких як низька текстурованість поверхонь, далекі об'єкти, складні умови освітлення або наявність динамічних об'єктів. Це робить датасет придатним

не лише для демонстрації переваг методів, але й для виявлення їх слабких сторін.

#### *1.4.5. Висновки до підрозділу*

Таким чином, еталонні датасети є невід’ємним інструментом дослідження систем комп’ютерного зору в автономних транспортних засобах. Серед них датасет KITTI займає особливе місце завдяки поєднанню стереоданих, точної калібровки та реалістичних дорожніх сцен. Саме це робить його доцільним вибором для експериментального дослідження геометричних та нейромережевих методів оцінки глибини сцени.

У наступному розділі буде розглянуто вибір програмних інструментів, бібліотек і технологій, які використовуються для практичної реалізації та експериментального дослідження системи комп’ютерного зору на основі розглянутих теоретичних положень.

### **1.5. Класифікація та порівняльний аналіз методів оцінки глибини сцени**

Оцінка глибини сцени та відстаней до об’єктів є багатовимірною задачею комп’ютерного зору, що поєднує геометричні, статистичні та навчальні підходи. У контексті навігації автономних транспортних засобів (АТЗ) ця задача набуває особливої значущості, оскільки від точності просторового сприйняття безпосередньо залежить безпека руху та коректність навігаційних рішень. У сучасних дослідженнях запропоновано значну кількість методів оцінки глибини, які відрізняються за принципами роботи, вимогами до вхідних даних та характеристиками отриманих результатів [8]–[12], [14].

Загалом методи оцінки глибини можна класифікувати за кількістю камер, наявністю апріорних знань, способом отримання масштабу та рівнем алгоритмічної складності. Така класифікація дозволяє не лише систематизувати існуючі підходи, але й обґрунтувати вибір конкретних рішень для практичних систем автономної навігації.

#### *1.5.1. Геометричні методи камерної оцінки глибини*



Геометричні методи базуються на строгих моделях формування зображення та використовують відомі параметри камерної системи. Найбільш поширеними серед них є стереоскопічні підходи, які застосовують дві або більше камер для відновлення просторової структури сцени. Основна ідея таких методів полягає у використанні геометричної відповідності між проєкціями одних і тих самих точок сцени на різних зображеннях.

Перевагою геометричних методів є їхня інтерпретованість та фізична обґрунтованість: отримані оцінки мають безпосередній зв'язок із реальними просторовими параметрами сцени. Саме тому такі підходи широко використовуються як еталонні у задачах оцінки якості інших методів, зокрема нейромережевих [23], [24]. Водночас ефективність геометричних алгоритмів суттєво залежить від якості калібрування камер, умов освітлення та текстурованості поверхонь, що обмежує їх застосування у складних дорожніх сценах [32].

#### *1.5.2. Монокулярні аналітичні підходи*

Монокулярні аналітичні методи використовують лише одне зображення та ґрунтуються на геометричних або евристичних припущеннях щодо структури сцени. До таких припущень належать знання про типові розміри об'єктів, їх розташування у просторі або закономірності перспективної проєкції. Подібні підходи є привабливими з точки зору простоти реалізації та низьких обчислювальних витрат.

Разом із тим, монокулярні аналітичні методи мають обмежену універсальність, оскільки їх точність безпосередньо залежить від коректності апріорних припущень. У реальних умовах дорожнього руху, де об'єкти можуть мати значну варіативність форм і розмірів, такі методи часто використовуються лише як допоміжні або попередні оцінки [17], [19].

#### *1.5.3. Нейромережеві методи монокулярної оцінки глибини*

Нейромережеві методи монокулярної оцінки глибини стали одним із найбільш інтенсивно досліджуваних напрямів у комп'ютерному зорі за останні роки. Ці підходи використовують глибокі згорткові нейронні мережі

або трансформерні архітектури для відновлення просторової структури сцени без явного використання геометричних відповідностей [8]–[12].

Основною перевагою MDE-підходів є їхня здатність узагальнювати складні просторові закономірності та працювати в умовах часткових завад, змінного освітлення або низької текстурованості сцени. Сучасні моделі демонструють високу якість відновлення відносної структури сцени, що робить їх перспективними для задач семантичної навігації та аналізу середовища [9], [15], [35].

Водночас характерною особливістю нейромережових монокулярних методів є відсутність абсолютного метричного масштабу. Це обмеження зумовлює необхідність додаткових процедур масштабування або калібрування результатів, зокрема із використанням стереоданих або еталонних об'єктів [27], [28].

#### *1.5.4. Порівняльна характеристика підходів*

З точки зору практичного застосування в АТЗ, різні підходи до оцінки глибини мають взаємодоповнювальний характер. Геометричні методи забезпечують метричну точність і можуть використовуватися як опорні, монокулярні аналітичні підходи - як швидкі та ресурсоефективні, а нейромережові методи - як універсальні засоби аналізу складних сцен, що показано у порівняльній Таблиці 1.1

*Таблиця 1.1*

#### **Порівняльна характеристика методів оцінки глибини**

<b>Критерій</b>	<b>Геометричні методи</b>	<b>Монокулярні аналітичні</b>	<b>Нейромережові MDE</b>
Кількість камер	Дві і більше	Одна	Одна
Метрична коректність	Висока	Умовна	Відсутня без калібрування
Стійкість до умов сцени	Середня	Низька	Висока
Обчислювальна складність	Середня	Низька	Висока
Придатність до реального часу	Висока	Висока	Обмежена

Аналіз наведених характеристик підтверджує доцільність комбінування різних підходів у рамках єдиної системи комп'ютерного зору. Такий підхід дозволяє компенсувати недоліки окремих методів і підвищити загальну надійність оцінки просторових параметрів сцени.

#### **1.6. Гібридні підходи до оцінки глибини та відстані в системах комп'ютерного зору**

У сучасних системах комп'ютерного зору для автономних транспортних засобів все частіше застосовуються гібридні підходи, які поєднують класичні геометричні методи з нейромережевими моделями. Така тенденція зумовлена тим, що жоден з підходів окремо не забезпечує одночасно високої точності, стабільності та універсальності в умовах реальної експлуатації АТЗ. Геометричні методи мають чітке фізичне обґрунтування та забезпечують метричну коректність, тоді як методи глибокого навчання демонструють високу здатність до узагальнення складних сцен, але часто страждають від проблеми масштабу та нестабільності оцінок [7], [8], [9].

Гібридні підходи передбачають використання різних джерел інформації про глибину сцени з подальшою їх агрегацією або взаємною калібруванням. У наукових дослідженнях зазначається, що поєднання стереоскопічних оцінок з монокулярними нейромережевими моделями дозволяє зменшити вплив шуму, компенсувати пропуски даних та підвищити стійкість системи до складних умов освітлення і середовища [23], [24], [33]. При цьому стерео-методи часто використовуються як джерело еталонної або квазі-еталонної інформації для корекції масштабу монокулярних оцінок.

Одним з поширених сценаріїв гібридного підходу є використання геометричних методів для калібрування або перевірки результатів нейромережових моделей. Наприклад, у контрольованих умовах або на окремих ділянках сцени, де стерео-оцінка є надійною, її результати можуть застосовуватися для масштабування або нормалізації монокулярної глибини. Такий підхід дозволяє поєднати фізичну коректність геометричних методів із

високою адаптивністю глибокого навчання, що підтверджується результатами сучасних досліджень у сфері монокулярної оцінки глибини [29], [35].

Іншим напрямом розвитку гібридних систем є ф'южн оцінок глибини на рівні програмної архітектури. У цьому випадку різні модулі комп'ютерного зору - стерео-оцінка, монокулярна глибина, оцінка відстані за відомими розмірами об'єктів або калібрувальними маркерами - розглядаються як незалежні джерела інформації. Їх результати агрегуються з урахуванням довіри до кожного методу, типу об'єкта та умов сцени. Подібні підходи відповідають сучасним принципам розробки модульних програмних систем, де окремі компоненти можуть бути замінені або вдосконалені без порушення цілісності рішення [1], [2], [4].

Важливу роль у гібридних системах відіграє також часовий аспект. Оцінки глибини та відстаней, отримані на окремих кадрах, можуть зазнавати локальних флуктуацій через шум сенсорів, помилки детекції або нестабільність нейромережових моделей. У зв'язку з цим у програмних реалізаціях часто застосовуються методи згладжування та агрегації по часу, що дозволяє підвищити стабільність оцінок без істотної втрати реактивності системи. Такі підходи узгоджуються з класичними методами фільтрації сигналів і широко використовуються в системах трекінгу та навігації [29], [30].

Слід зазначити, що ефективність гібридних підходів значною мірою залежить від програмної реалізації та організації обчислювального процесу. Поєднання декількох методів збільшує обчислювальне навантаження, що вимагає оптимізації алгоритмів та раціонального розподілу ресурсів. У цьому контексті особливого значення набуває використання сучасних бібліотек комп'ютерного зору та фреймворків глибокого навчання, які забезпечують апаратне прискорення та підтримку паралельних обчислень [21], [25].

Таким чином, гібридні підходи до оцінки глибини та відстані в системах комп'ютерного зору дозволяють поєднати переваги геометричних і нейромережових методів, компенсуючи їхні індивідуальні обмеження. У контексті автономних транспортних засобів такі підходи розглядаються як

найбільш перспективні, оскільки забезпечують баланс між точністю, стабільністю та обчислювальною ефективністю. Подальше дослідження та експериментальна перевірка гібридних методів потребують ретельного підбору інструментарію та реалізації програмних прототипів, що є предметом наступного розділу роботи.

### **Висновки до розділу 1**

У першому розділі магістерської роботи здійснено комплексний теоретичний аналіз сучасних підходів до побудови систем комп'ютерного зору для задач навігації автономних транспортних засобів. Розглянуто роль комп'ютерного зору як ключової програмно-алгоритмічної підсистеми АТЗ, що забезпечує сприйняття навколишнього середовища, інтерпретацію дорожньої сцени та формування вхідних даних для модулів планування і керування рухом.

Показано, що візуальна інформація, отримана з камер, є одним з найбільш інформативних і доступних джерел даних для автономної навігації. Проаналізовано основні функціональні завдання систем комп'ютерного зору, зокрема детекцію та класифікацію об'єктів, аналіз сцен дорожнього руху, а також оцінку просторових характеристик середовища, що є критично важливими для безпечного функціонування автономних транспортних засобів.

У розділі розглянуто фундаментальні геометричні принципи формування зображення та роль калібрування камер у забезпеченні коректності просторових оцінок. Проаналізовано класичні підходи до оцінки глибини сцени, зокрема бінокулярні (стереоскопічні) методи, які мають чітке фізичне обґрунтування та забезпечують метричну точність, але водночас залежать від умов текстурованості сцени та якості вхідних даних.

Окрему увагу приділено сучасним неймережевим методам оцінки глибини на основі монокулярних зображень. Показано, що такі підходи здатні ефективно узагальнювати складні просторові структури сцени, однак мають обмеження, пов'язані з відносним характером оцінок глибини та

нестабільністю масштабу. Аналіз наукових джерел свідчить, що жоден з підходів - геометричний або нейромережевий - не є універсальним для всіх умов експлуатації автономних систем.

На основі проведеного аналізу обґрунтовано доцільність застосування гібридних підходів, які поєднують переваги класичних методів комп'ютерного зору та глибокого навчання. Показано, що такі підходи дозволяють підвищити стабільність та надійність оцінки глибини і відстаней до об'єктів за рахунок використання декількох джерел інформації та їх програмної агрегації. Особливу роль у цьому відіграють принципи модульної програмної архітектури та оптимізації обчислювальних процесів, що є характерними для сучасних систем інженерії програмного забезпечення.

Отримані в першому розділі теоретичні положення формують наукову та методологічну основу для подальшого дослідження. Вони визначають вимоги до інструментів, алгоритмів та програмних рішень, які будуть використані для побудови та експериментальної оцінки системи комп'ютерного зору. У наступному розділі на основі сформованих теоретичних висновків буде здійснено вибір програмного інструментарію, датасетів та методів реалізації, а також описано практичну архітектуру системи комп'ютерного зору для задач навігації автономного транспортного засобу.

## РОЗДІЛ 2

### ВИБІР ІНСТРУМЕНТІВ ТА ПРОГРАМНОГО СЕРЕДОВИЩА ДЛЯ РОЗРОБКИ СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ

#### 2.1. Загальні вимоги до програмного середовища системи комп'ютерного зору АТЗ

Реалізація системи комп'ютерного зору для задач навігації автономного транспортного засобу (АТЗ) висуває підвищені вимоги до програмного середовища, інструментів розробки та бібліотек обробки даних. На відміну від традиційних прикладних інформаційних систем, програмні компоненти комп'ютерного зору працюють з великими обсягами візуальної інформації, потребують значних обчислювальних ресурсів та повинні забезпечувати обробку даних у режимі, близькому до реального часу.

Основними вимогами до програмного середовища системи комп'ютерного зору АТЗ є:

- висока продуктивність обробки зображень і відеопотоків, що забезпечує своєчасне формування навігаційних рішень;
- масштабованість та модульність архітектури, які дозволяють розширювати або замінювати окремі алгоритмічні компоненти без суттєвих змін у системі;
- підтримка класичних і нейромережових методів комп'ютерного зору, з можливістю їх комбінування;
- сумісність з відкритими стандартами та датасетами, що є важливим для відтворюваності експериментів і порівняння результатів;
- можливість апаратного прискорення, зокрема використання GPU для паралельних обчислень.

У сучасних наукових дослідженнях з комп'ютерного зору та інженерії програмного забезпечення підкреслюється, що вибір програмних інструментів істотно впливає не лише на швидкодію системи, але й на якість реалізації алгоритмів, зручність налагодження та можливість подальшого розвитку програмного продукту [6], [14]. Тому на етапі проєктування системи

комп'ютерного зору доцільно використовувати перевірені програмні платформи, які поєднують ефективність, гнучкість та широку підтримку науковою спільнотою.

З урахуванням результатів теоретичного аналізу, виконаного в Розділі 1, програмне середовище системи комп'ютерного зору має забезпечувати реалізацію таких функціональних компонентів:

- зчитування та попередню обробку зображень з камер;
- реалізацію геометричних методів комп'ютерного зору (калібрування, ректифікація, оцінка глибини);
- застосування нейромережових моделей для детекції об'єктів та монокулярної оцінки глибини;
- інтеграцію з датасетами для навчання та тестування алгоритмів;
- візуалізацію результатів та збір експериментальних метрик.

У зв'язку з цим у межах даної роботи здійснено обґрунтований вибір програмних інструментів і бібліотек, які забезпечують реалізацію повного ланцюга обробки візуальних даних у системі комп'ютерного зору АТЗ — від низькорівневої обробки зображень до оцінки просторових характеристик сцени та кількісного аналізу точності отриманих результатів. Подальші підрозділи цього розділу присвячено обґрунтуванню вибору конкретних бібліотек, фреймворків і наборів даних, а також визначенню їх ролі у побудові та дослідженні розроблюваної системи.

## **2.2. Загальна схема інструментів і програмного середовища системи комп'ютерного зору**

### *2.2.1. Загальна архітектура інструментів*

Для реалізації системи комп'ютерного зору для задач навігації автономного транспортного засобу використано поєднання класичних бібліотек обробки зображень, фреймворків глибокого навчання та спеціалізованих датасетів. Такий підхід дозволяє поєднати геометричні



методи комп'ютерного зору з неймережевими моделями, забезпечуючи гнучкість та відтворюваність експериментів.

Програмне середовище системи побудовано за модульним принципом, у межах якого кожен інструмент відповідає за окремий етап обробки та аналізу візуальних даних: зчитування та підготовку даних, аналіз зображень, оцінку просторових характеристик сцени та візуалізацію результатів, як показано на рисунку 2.1.

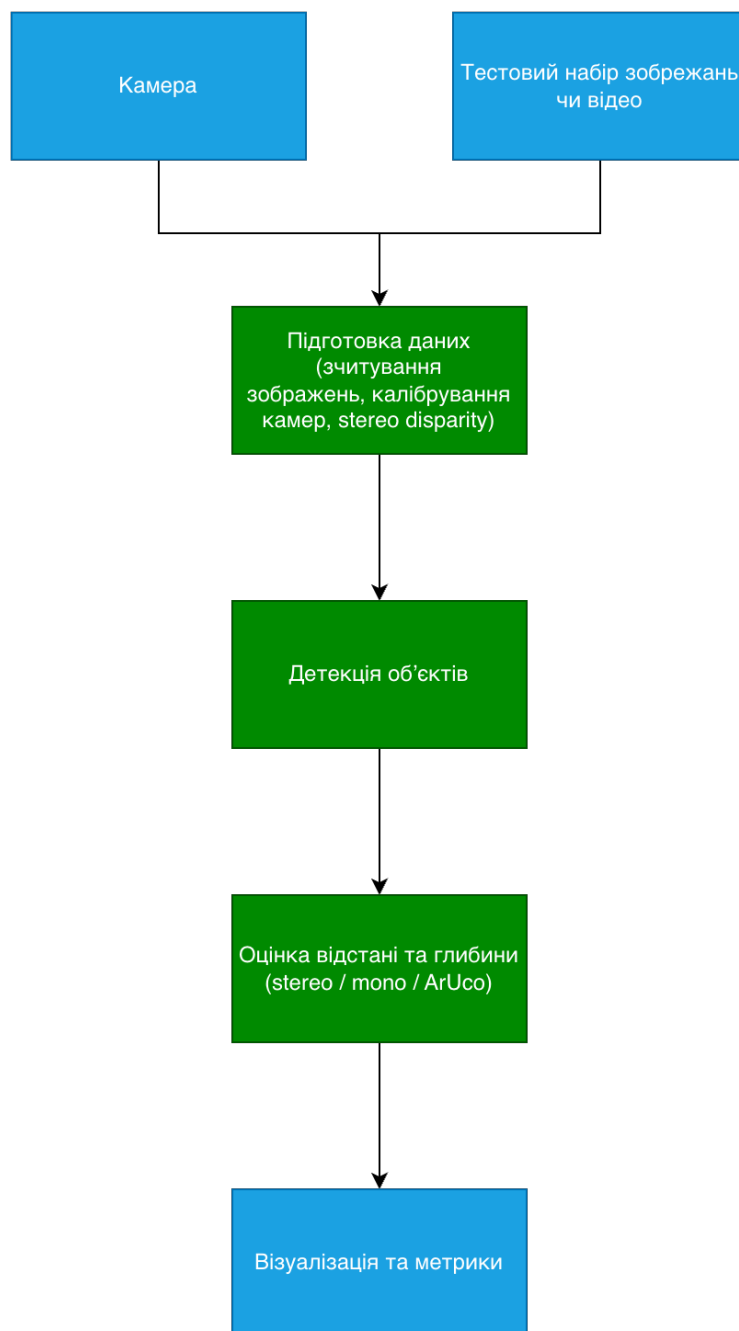


Рисунок 2.1. Алгоритм обробки даних

### 2.2.2. Вибір програмних інструментів та середовища реалізації

Для реалізації системи комп'ютерного зору в задачах навігації автономного транспортного засобу було обрано набір програмних інструментів, які забезпечують повний цикл обробки візуальних даних — від зчитування та попередньої обробки зображень до оцінки просторових характеристик сцени та аналізу результатів. Вибір інструментів здійснювався з урахуванням вимог до продуктивності, масштабованості, відтворюваності експериментів та можливості інтеграції класичних алгоритмів комп'ютерного зору з методами глибокого навчання.

У таблиці 2.1 наведено перелік основних програмних засобів, використаних у роботі, а також їх функціональне призначення та обґрунтування вибору. Подібна структуризація інструментів дозволяє чітко визначити роль кожного компонента в загальній архітектурі системи та слугує основою для подальшого детального розгляду окремих бібліотек і фреймворків у наступних підрозділах.

Таблиця 2.1

#### Обрані інструменти та їх призначення

Інструмент	Призначення у системі	Обґрунтування вибору
OpenCV	Попередня обробка зображень, калібрування, ректифікація, stereo	Висока продуктивність, підтримка класичних CV-алгоритмів
TensorFlow	Нейромережеві моделі детекції та моноглибини	Масштабованість, підтримка pretrained моделей
KITTI	Навчання та тестування алгоритмів	Наявність GT-глибини та калібрувань
Python	Реалізація пайплайну та експериментів	Швидка розробка, багата екосистема
GPU (опційно)	Прискорення нейромережевих обчислень	Реальний час обробки

### 2.3. Використання бібліотеки OpenCV у задачах комп'ютерного зору

Бібліотека OpenCV (Open Source Computer Vision Library) є одним із базових програмних інструментів для реалізації систем комп'ютерного зору в задачах автономної навігації. Вона надає широкий набір алгоритмів для

обробки зображень, геометричних перетворень, калібрування камер, стереобачення та аналізу відеопотоків у реальному часі. Завдяки відкритому коду, високій продуктивності та активній підтримці спільноти OpenCV широко використовується як у наукових дослідженнях, так і в промислових програмних рішеннях [25].

У межах даного дослідження бібліотека OpenCV застосовується як базовий програмний інструмент для реалізації низькорівневих та середньорівневих етапів обробки візуальних даних автономного транспортного засобу, включаючи попередню обробку зображень, геометричні перетворення та підготовку даних для подальшого аналізу. До таких етапів належать зчитування та попередня обробка зображень, калібрування та корекція дисторсій камер, ректіфікація стереопар, побудова карти диспаритету, а також візуалізація результатів обробки. Застосування OpenCV дозволяє відокремити класичні геометричні алгоритми від нейромережових компонентів, що спрощує архітектуру системи та підвищує її модульність.

Однією з ключових переваг OpenCV є наявність реалізацій математично обґрунтованих методів, які мають чітку фізичну інтерпретацію. Зокрема, бібліотека підтримує повний цикл роботи з камерною моделлю, включаючи зчитування параметрів внутрішньої та зовнішньої калібровки, побудову матриці камери та виконання корекції оптичних спотворень на основі відомих моделей [26]. Це є критично важливим для задач оцінки відстаней, оскільки точність подальших обчислень безпосередньо залежить від коректності геометричної моделі камери.

У задачах бінокулярного зору OpenCV надає ефективні алгоритми для обчислення карти диспаритету, зокрема StereoBM та StereoSGBM, які широко застосовуються в системах автономної навігації [24]. Алгоритми стереозіставлення дозволяють отримати щільну карту відносної глибини сцени на основі ректіфікованих зображень з лівої та правої камер. Отримані карти диспаритету надалі використовуються для оцінки метричної глибини та

відстаней до об'єктів, що є одним із ключових завдань комп'ютерного зору в АТЗ.

Крім стереоалгоритмів, OpenCV містить інструменти для роботи з маркерами та ключовими точками, які можуть бути використані для калібрувальних та контрольних експериментів. Зокрема, модуль `cv2.aruco` дозволяє детектувати маркери ArUco та оцінювати їх положення відносно камери за допомогою методів розв'язання задачі визначення пози (PnP) [27]. Такі можливості є корисними для перевірки точності геометричних моделей і масштабування результатів нейромережових оцінок глибини.

Важливою особливістю OpenCV є її орієнтація на роботу в реальному часі. Бібліотека оптимізована для виконання на центральних та графічних процесорах і підтримує багатопотокову обробку даних. Це дозволяє використовувати OpenCV у складі різноманітних модулів автономних систем, де затримка обробки кадру безпосередньо впливає на безпеку та якість навігаційних рішень [6].

У межах даної роботи OpenCV виконує роль фундаментального шару комп'ютерного зору, поверх якого інтегруються нейромережові моделі, реалізовані за допомогою TensorFlow. Такий підхід дозволяє поєднати переваги класичних геометричних методів із гнучкістю та узагальнюючими можливостями глибокого навчання. Взаємодія OpenCV і TensorFlow забезпечує повний цикл обробки візуальних даних — від сирих зображень до оцінки просторових характеристик сцени, що є необхідною умовою для побудови ефективних систем навігації автономних транспортних засобів.

## **2.4. Використання платформи TensorFlow та методів глибокого навчання в системах комп'ютерного зору**

### *2.4.1. Загальна огляд платформи TensorFlow*

Сучасні задачі комп'ютерного зору в системах навігації автономних транспортних засобів характеризуються високою складністю та різноманітністю візуальних сцен, що ускладнює їх розв'язання виключно за допомогою класичних алгоритмів обробки зображень. У зв'язку з цим

значного поширення набули методи глибокого навчання, здатні автоматично формувати багаторівневі представлення візуальних даних та узагальнювати складні закономірності у зображеннях. Однією з найбільш поширених програмних платформ для реалізації таких методів є TensorFlow.

TensorFlow є відкритою програмною бібліотекою для побудови та виконання моделей машинного і глибокого навчання, що забезпечує підтримку широкого спектра нейромережових архітектур, ефективну роботу з багатовимірними тензорами та можливість масштабування обчислень на різні апаратні платформи. У наукових дослідженнях TensorFlow розглядається як універсальне середовище для створення, навчання та розгортання моделей комп'ютерного зору, зокрема у задачах детекції об'єктів, сегментації сцен та оцінки глибини.

Однією з ключових переваг використання TensorFlow у системах автономної навігації є можливість інтеграції нейромережових моделей у програмні пайплайни реального часу. Платформа підтримує оптимізовані обчислювальні графи, апаратне прискорення за допомогою GPU та ефективну роботу з потоками даних, що є критично важливим для обробки відеопослідовностей з камер АТЗ. Крім того, модульна архітектура TensorFlow спрощує поєднання нейромережових компонентів із класичними алгоритмами, реалізованими, зокрема, за допомогою бібліотеки OpenCV.

У контексті задач комп'ютерного зору методи глибокого навчання в TensorFlow застосовуються для розв'язання двох взаємопов'язаних проблем: семантичного аналізу сцени та оцінки просторових характеристик об'єктів. До першої групи належать задачі детекції та класифікації об'єктів дорожньої інфраструктури, які забезпечують ідентифікацію автомобілів, пішоходів, дорожніх знаків і розмітки. До другої — задачі оцінки глибини сцени та відстаней до об'єктів, що є необхідними для прийняття навігаційних рішень.

Особливе місце серед нейромережових підходів займають методи монокулярної оцінки глибини (Monocular Depth Estimation, MDE), які дозволяють відновлювати відносну або метричну глибину сцени за одним

зображенням. Такі моделі навчаються на великих наборах даних та здатні формувати щільні карти глибини, що відображають просторову структуру сцени. У порівнянні зі стереоскопічними методами, MDE не потребує наявності двох синхронізованих камер, однак вимагає додаткових процедур масштабування та калібрування для отримання метричних оцінок відстаней.

Важливим аспектом використання TensorFlow є підтримка попередньо навчених моделей та можливість їх адаптації до конкретних задач. Це дозволяє зменшити обчислювальні витрати на навчання та зосередитися на інтеграції моделей у прикладну систему. У системах автономної навігації така стратегія є доцільною, оскільки дає змогу поєднувати результати нейромережевого аналізу з геометричними методами оцінки відстаней, підвищуючи загальну надійність системи.

З програмної точки зору використання TensorFlow передбачає чітке розділення етапів інференсу нейромережових моделей та подальшої обробки результатів. Вихідні дані моделей глибокого навчання, зокрема карти глибини або bounding box об'єктів, передаються до модулів постобробки, де за допомогою класичних методів виконується агрегація, фільтрація та приведення результатів до зручного для навігаційних підсистем формату. Такий підхід відповідає принципам модульної архітектури та спрощує експериментальне порівняння різних алгоритмічних рішень.

Отже, застосування платформи TensorFlow у поєднанні з методами глибокого навчання дозволяє суттєво розширити можливості систем комп'ютерного зору для автономних транспортних засобів. Нейромережеві моделі доповнюють класичні геометричні підходи, забезпечуючи підвищену стійкість до складних умов середовища та багатства сцен. У наступному підрозділі буде розглянуто практичні аспекти застосування нейромережових методів оцінки глибини сцени та їх інтеграції з результатами класичних алгоритмів.

#### *2.4.2. Монокулярна оцінка глибини (Monocular Depth Estimation)*

Монокулярна оцінка глибини (Monocular Depth Estimation, MDE) є одним із ключових напрямів розвитку сучасних систем комп'ютерного зору для навігації автономних транспортних засобів (АТЗ). На відміну від стереоскопічних методів, що використовують пару синхронізованих камер, MDE базується на аналізі одного зображення та дозволяє відновлювати просторову структуру сцени без додаткових апаратних засобів. Такий підхід є привабливим з точки зору зниження апаратної складності системи та спрощення її інтеграції в програмну архітектуру АТЗ.

У традиційних геометричних підходах оцінка глибини сцени з одного зображення вважається некоректно поставленою задачею, оскільки інформація про масштаб сцени втрачається під час проєкції тривимірного простору на двовимірну площину. Проте сучасні методи глибокого навчання дозволяють частково компенсувати цю проблему за рахунок навчання на великих датасетах, де нейромережеві моделі засвоюють статистичні закономірності між візуальними ознаками та реальними відстанями [8]–[12].

У більшості сучасних MDE-підходів використовується згортова нейронна мережа або трансформерна архітектура, яка перетворює вхідне RGB-зображення у щільну карту глибини. При цьому кожному пікселю ставиться у відповідність значення відносної або абсолютної глибини. Огляд сучасних методів монокулярної оцінки глибини показує, що найбільш ефективними є моделі, здатні поєднувати локальні текстурні ознаки з глобальним контекстом сцени [8], [9], [15].

Залежно від способу навчання розрізняють супервізовані, напівсупервізовані та самонавчальні (self-supervised) підходи до MDE. У супервізованих моделях як еталонні дані використовуються карти глибини, отримані за допомогою лідара або стереосистем, що забезпечує високу точність, але обмежує доступність навчальних даних. Самонавчальні методи, навпаки, дозволяють навчати моделі без явної глибини, використовуючи фотометричну узгодженість між послідовними кадрами або стереопарами, що широко застосовується в задачах автономного водіння [27], [28].

Важливим обмеженням MDE є проблема масштабної неоднозначності. У багатьох випадках мережа коректно відтворює відносну структуру сцени, проте абсолютні значення глибини можуть змінюватися між різними сценами або умовами зйомки. Для розв'язання цієї проблеми запропоновано підходи з відновленням масштабу, а також комбінування MDE з геометричними або сенсорними методами [19], [29].

У контексті автономних транспортних засобів монокулярна оцінка глибини зазвичай використовується не ізольовано, а в поєднанні з іншими підсистемами комп'ютерного зору. Зокрема, карта глибини може інтегруватися з результатами детекції об'єктів для оцінки відстаней до транспортних засобів і пішоходів, а також застосовуватися як додаткове джерело інформації у випадках деградації стереозору або відсутності коректних диспаратних даних [14], [15].

З практичної точки зору реалізація MDE у програмних системах АТЗ ґрунтується на використанні спеціалізованих фреймворків глибокого навчання, таких як TensorFlow, що забезпечують апаратне прискорення обчислень та можливість масштабування моделей. Попередньо навчені моделі, зокрема сімейства MiDaS або AdaBins, дозволяють інтегрувати MDE у прикладні системи без необхідності повного циклу навчання [22], [29].

Таким чином, монокулярна оцінка глибини є важливим компонентом сучасних систем комп'ютерного зору для автономної навігації, забезпечуючи компактне та гнучке рішення для аналізу просторової структури сцени. Водночас обмеження, пов'язані з точністю абсолютних оцінок та залежністю від умов зйомки, зумовлюють доцільність її використання у поєднанні з геометричними методами, що буде розглянуто в наступних підрозділах.

#### *2.4.3. Обмеження та джерела похибок монокулярної оцінки глибини в реальних сценах*

Незважаючи на значний прогрес у розвитку нейромережевих методів монокулярної оцінки глибини, практичне застосування таких підходів у системах навігації автономних транспортних засобів (АТЗ) супроводжується



низкою обмежень та характерних джерел похибок. Ці обмеження зумовлені як фізичними властивостями процесу формування зображення, так і особливостями навчання та узагальнення нейромережових моделей.

Ключовою фундаментальною проблемою MDE є відсутність абсолютної метричної шкали. Оскільки модель отримує на вхід лише одне зображення, вона не має прямої інформації про реальні розміри об'єктів та геометричний базис зйомки. У результаті більшість нейромережових моделей відновлюють лише відносну глибину сцени, що проявляється у глобальних зсувах масштабу між різними сценами або навіть між окремими кадрами одного відеопотоку. Подібний ефект особливо помітний у сценах з нестандартною перспективою або нетиповими пропорціями об'єктів [8], [9], [12].

Іншим суттєвим джерелом похибок є залежність якості оцінки глибини від умов освітлення та контрастності сцени. У нічних умовах, при наявності туману, дощу або сильних відблисків, візуальні ознаки, на яких базується нейромережа, стають менш інформативними. У таких випадках модель може неправильно інтерпретувати градієнти яскравості як зміну глибини або, навпаки, ігнорувати реальні просторові переходи [14], [31]. Для автономних систем це створює додаткові ризики, оскільки помилки в оцінці відстаней безпосередньо впливають на безпеку руху.

Значний вплив на точність MDE мають також особливості сцен з низькою текстурною насиченістю. Однорідні поверхні, такі як дорожнє полотно, стіни будівель або небо, містять мінімальну кількість локальних ознак, що ускладнює коректне відновлення глибини. У таких випадках нейромережева модель часто спирається на контекстні припущення, які можуть бути некоректними для конкретної сцени, що призводить до локальних або систематичних помилок [10], [11].

Окрему категорію проблемних сценаріїв становлять тонкі або дрібні об'єкти, зокрема дорожні знаки, стовпи освітлення та елементи огорожень. Через обмежену просторову роздільну здатність глибини та згладжувальний характер нейромережових моделей такі об'єкти можуть бути частково або

повністю втрачені на карті глибини. Це особливо критично для задач навігації, де навіть невеликі перешкоди можуть мати значення [16], [18].

Ще одним джерелом похибок є обмежена здатність моделей MDE коректно працювати з прозорими або відбивними поверхнями, такими як скло, дзеркала або мокрий асфальт. У подібних випадках візуальна інформація не відповідає реальній геометрії сцени, що вводить нейромережу в оману та призводить до некоректних оцінок глибини або появи артефактів на карті [31].

Не менш важливим чинником є обмеження, пов'язані з навчальними даними. Більшість моделей MDE тренуються на великих датасетах, зокрема KITTI, які хоча й охоплюють широкий спектр дорожніх сценаріїв, але не здатні повністю відобразити різноманіття реальних умов експлуатації. Це призводить до зниження точності при перенесенні моделей на нові середовища, інші камери або відмінні параметри зйомки [23], [27].

Таким чином, монокулярна оцінка глибини характеризується низкою системних обмежень, пов'язаних із відсутністю явної метричної шкали та високою чутливістю до умов сцени. Це зумовлює доцільність її застосування не як самостійного, а як допоміжного компонента системи оцінки просторових параметрів. У контексті автономних транспортних засобів методи MDE доцільно розглядати у складі багатокomпонентної системи аналізу візуальних даних, де отримані оцінки можуть бути уточнені або стабілізовані шляхом поєднання з геометричними підходами, калібрувальними методами, а також процедурами ф'южну та згладжування, розгляд яких наведено у наступному підрозділі.

#### **2.4.4. Ф'южн та згладжування оцінок глибини**

З огляду на обмеження окремих підходів до оцінки глибини сцени, у сучасних системах комп'ютерного зору для автономних транспортних засобів (АТЗ) все більшого поширення набувають методи ф'южну та згладжування оцінок. Їх метою є підвищення стабільності, надійності та метричної точності визначення відстаней шляхом комбінування результатів різних алгоритмічних підходів і обробки послідовних вимірювань у часі.

Ф'южн оцінок глибини передбачає об'єднання інформації, отриманої з декількох джерел або методів, зокрема геометричних (стереозір, pinhole-модель, ArUco-маркери) та нейромережових (монокулярна оцінка глибини, MDE). Кожен із цих підходів має власні сильні та слабкі сторони: геометричні методи забезпечують високу метричну точність у контрольованих умовах, тоді як нейромережові моделі демонструють кращу узагальнювальну здатність у складних сценах. Поєднання таких оцінок дозволяє компенсувати недоліки окремих методів і підвищити загальну якість результату [14], [18].

На практиці ф'южн може реалізовуватися у вигляді зваженого усереднення оцінок глибини, де вагові коефіцієнти визначаються залежно від типу сцени, діапазону відстаней або рівня довіри до конкретного методу. Наприклад, у ближній зоні перевага може надаватися геометричним методам, тоді як для віддалених об'єктів доцільно використовувати інформацію з монокулярних нейромережових моделей. Альтернативним підходом є селективний вибір методу оцінки глибини залежно від класу об'єкта або умов спостереження.

Окремим важливим аспектом є просторове згладжування оцінок глибини. У задачах оцінки відстані до об'єктів часто використовується агрегація значень глибини в межах області інтересу (Region of Interest, ROI), що відповідає bounding box детектованого об'єкта. Застосування надійних статистичних показників, таких як медіана або квантілі, дозволяє зменшити вплив шуму, локальних артефактів та поодиноких хибних значень, характерних як для стерео-, так і для нейромережових карт глибини.

Часове згладжування оцінок глибини є не менш важливим для систем автономної навігації, оскільки поодинокі кадри можуть містити значні флуктуації вимірювань. Для стабілізації оцінок у послідовності кадрів застосовуються методи експоненційного згладжування або фільтра Калмана. Такі підходи дозволяють зменшити стрибки значень глибини, зберігаючи при цьому чутливість системи до реальних змін положення об'єктів у просторі [29].

Поєднання ф'южну та згладжування формує багаторівневий механізм підвищення якості оцінки глибини, який є особливо актуальним у реальних умовах експлуатації АТЗ. Застосування цих методів дозволяє перейти від нестабільних сирих оцінок до більш надійних і інтерпретованих значень відстаней, придатних для використання у модулях планування руху та керування.

Таким чином, ф'южн та згладжування оцінок глибини виступають невід'ємними складовими сучасних систем штучного зору автономних транспортних засобів. Вони забезпечують підвищення точності та стійкості роботи систем комп'ютерного зору і створюють основу для коректного порівняльного аналізу різних методів оцінки глибини, який розглядається у наступному розділі роботи.

## **2.5. Порівняльний аналіз методів оцінки глибини, використаних у дослідженні**

У межах даної магістерської роботи порівняльний аналіз методів оцінки глибини сцени виконується з урахуванням конкретних технологій, бібліотек та програмних рішень, обраних для реалізації експериментального пайплайну. Основна увага приділяється практичній придатності методів у контексті навігації автономних транспортних засобів (АТЗ), їхній точності, стабільності та обчислювальній ефективності.

У дослідженні реалізовано та проаналізовано такі підходи:

- бінокулярну (стерео) оцінку глибини на основі алгоритмів StereoBM / StereoSGBM з бібліотеки OpenCV;
- монокулярну геометричну оцінку відстані на основі pinhole-моделі та детекції об'єктів;
- монокулярну нейромережеву оцінку глибини (MDE) з використанням моделей MiDaS у середовищі TensorFlow;
- еталонну оцінку відстані за допомогою ArUco-маркерів і розв'язання задачі PnP.

Бінокулярний підхід, реалізований із використанням калібрувальних параметрів датасету KITTI, слугує базовим геометричним методом у даному дослідженні. Його ключовою перевагою є можливість отримання метрично коректної карти глибини без використання навчальних вибірок. Практичні експерименти показали, що стерео-метод забезпечує стабільні оцінки відстані до об'єктів за умови достатньої текстурованості сцени та коректної ректифікації зображень. Разом із тим, його ефективність суттєво знижується на однорідних або блискучих поверхнях, що проявляється у появі пропусків або шуму на карті диспаратів.

Монокулярний pinhole-підхід, реалізований у поєднанні з детектором об'єктів (SSD MobileNet), характеризується низькою обчислювальною складністю та високою швидкістю. Однак результати експериментів підтверджують, що точність такого методу значною мірою залежить від правильності класифікації об'єкта та припущень щодо його реальних геометричних розмірів. У практичних сценаріях це призводить до зростання похибки на віддалених об'єктах та у випадках нетипових пропорцій.

Монокулярна нейромережева оцінка глибини (MDE), реалізована за допомогою моделі MiDaS\_small, продемонструвала здатність коректно відновлювати просторову структуру сцени навіть у складних умовах. Водночас експерименти на датасеті KITTI показали, що MDE забезпечує лише відносну глибину і потребує додаткового масштабування для отримання метричних значень. Незважаючи на це, нейромережевий підхід виявився менш чутливим до відсутності текстур та локальних дефектів зображення порівняно зі стерео-методами, що робить його корисним доповненням у комбінованих системах.

Метод оцінки відстані на основі ArUco-маркерів і задачі визначення пози (PnP) використовується в роботі як еталонний. Він забезпечує найвищу точність метричних оцінок, проте має суттєві обмеження щодо сфери застосування, оскільки потребує спеціально підготовлених сцен. У контексті даного

дослідження ArUco-метод використовується для калібрування, перевірки правильності інших підходів та оцінки їх похибок.

З точки зору обчислювальної ефективності, реалізовані методи демонструють суттєві відмінності. Геометричні підходи на базі OpenCV забезпечують роботу в режимі, близькому до реального часу, тоді як нейромережеві MDE-моделі характеризуються значно більшою затримкою при виконанні на CPU. Це підкреслює важливість вибору апаратної платформи та необхідність оптимізації при практичному впровадженні таких рішень у системи АТЗ.

Таким чином, результати порівняльного аналізу реалізованих у роботі методів підтверджують доцільність використання комбінованого підходу, у якому стерео-оцінка з OpenCV виступає базовим джерелом метричної глибини, нейромережеві MDE-моделі використовуються для підвищення стійкості у складних сценах, а pinhole- та ArUco-методи застосовуються як допоміжні або еталонні. Така інтеграція обраних технологій створює основу для побудови надійної системи оцінки відстаней у задачах навігації автономних транспортних засобів та логічно підводить до експериментального аналізу, представленого у наступному розділі.

## **Висновки до розділу 2**

У другому розділі магістерської роботи виконано обґрунтований вибір програмних інструментів, бібліотек та технологій, необхідних для розробки та дослідження системи комп'ютерного зору для задач навігації автономних транспортних засобів. Розглянуто сучасні підходи до реалізації алгоритмів обробки зображень і оцінки глибини сцени з урахуванням вимог реального часу, точності та масштабованості програмних рішень.

У розділі проаналізовано можливості бібліотеки OpenCV як базового інструменту для реалізації геометричних методів комп'ютерного зору, зокрема задач калібрування камер, ректифікації стереопар, обчислення карти диспаратів та оцінки метричної глибини сцени. Показано, що використання класичних алгоритмів стереозору дозволяє отримувати стабільні та фізично

обґрунтовані оцінки відстаней у структурованих сценах, що робить їх придатними для навігаційних задач.

Окрему увагу приділено застосуванню нейромережевих методів глибокого навчання з використанням фреймворку TensorFlow. Розглянуто архітектури монокулярної оцінки глибини, включаючи згорткові нейронні мережі та моделі на основі Vision Transformers, а також проаналізовано їх переваги й обмеження в умовах реальних дорожніх сцен. Показано, що нейромережеві підходи забезпечують високу якість відносної глибини сцени, однак потребують додаткових процедур масштабування для отримання метричних значень.

У розділі також розглянуто методи допоміжної та еталонної оцінки відстаней, зокрема використання ArUco-маркерів і розв'язання задачі визначення пози (PnP), а також підходи до ф'южну та згладжування оцінок глибини. Обґрунтовано доцільність поєднання різних методів оцінки глибини з метою підвищення стабільності та надійності результатів у складних умовах експлуатації.

Результати проведеного аналізу дозволили сформулювати програмно-алгоритмічний пайплайн системи комп'ютерного зору, який поєднує геометричні та нейромережеві підходи й базується на використанні датасету KITTI для калібрування та верифікації результатів. Обрані інструменти та технології створюють практичну основу для реалізації експериментальної частини дослідження та проведення порівняльного аналізу точності, продуктивності й обмежень розглянутих методів, що є предметом наступного розділу магістерської роботи

### **РОЗДІЛ 3**

## **ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ВИМІРЮВАННЯ ВІДСТАНІ ДО ОБ'ЄКТІВ У СИСТЕМАХ КОМП'ЮТЕРНОГО ЗОРУ АТЗ**

### **3.1. Мета та постановка експериментальних досліджень**

Метою експериментальних досліджень, проведених у межах даної магістерської роботи, є кількісна та якісна оцінка ефективності методів вимірювання відстані до об'єктів у системах комп'ютерного зору автономних транспортних засобів (АТЗ). На відміну від класичних задач реконструкції сцени, у яких основна увага приділяється відновленню карти глибини, у даній роботі ключовим результатом є метрично коректна оцінка відстані до детектованих об'єктів, що безпосередньо використовується в задачах навігації, планування руху та забезпечення безпеки.

Експериментальні дослідження спрямовані на аналіз і порівняння кількох підходів до вимірювання відстані, які ґрунтуються на різних принципах обробки візуальної інформації. Зокрема, розглядаються бінокулярні (стереоскопічні) методи, що використовують геометричні співвідношення між диспаратетом і відстанню, монокулярні геометричні методи, засновані на *pinhole*-моделі камери та апіорних припущеннях щодо розмірів об'єктів, а також монокулярні нейромережеві методи, у яких карта глибини використовується як проміжне представлення для подальшого обчислення відстані.

Загальна логіка експериментального процесу вимірювання відстані, починаючи від обробки вхідних зображень і завершуючи обчисленням метричних похибок, узагальнено представлена на рисунку 3.1, який ілюструє послідовність етапів експериментального пайплайну та місце кожного з досліджуваних методів у загальній структурі обробки даних.



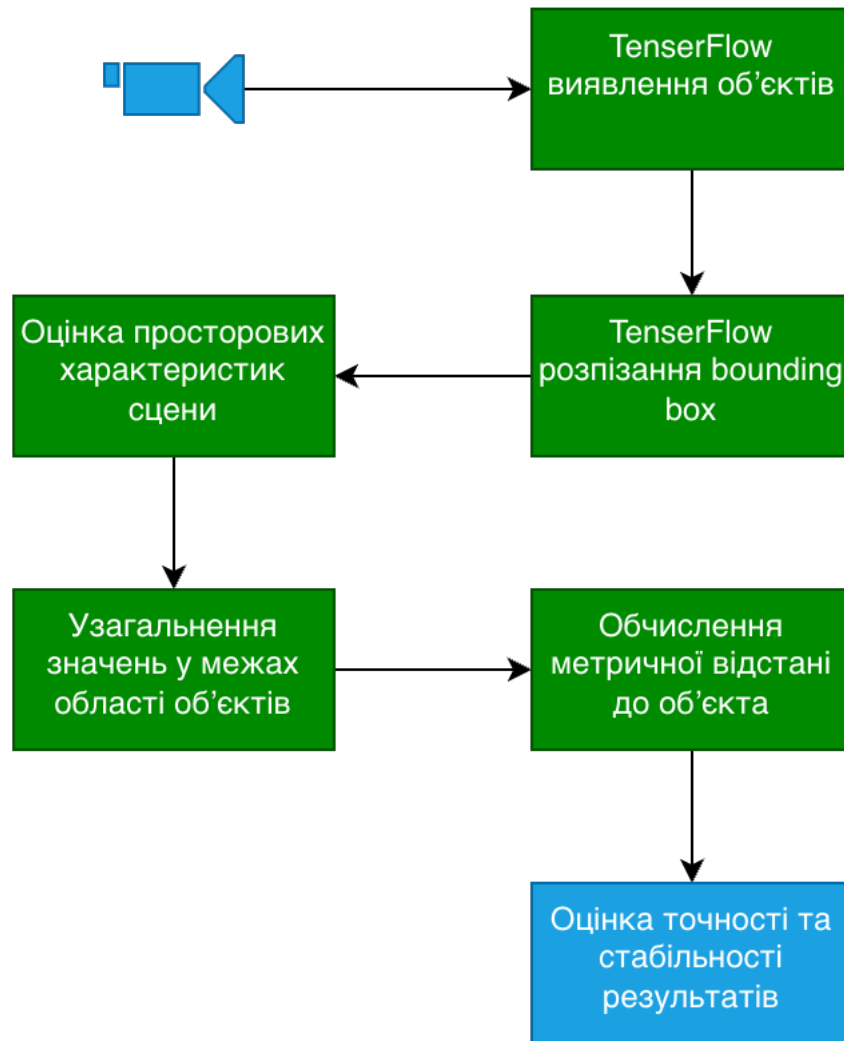


Рисунок 3.1. Схема експериментального пайплайну

Постановка експериментів передбачає використання датасету KITTI, який надає калібровані стереопари зображень та еталонні карти диспаритету, що дозволяє обчислювати реальні відстані до об'єктів у сцені. Бінокулярні оцінки відстані, отримані на основі стереоданих, використовуються як базова точка порівняння для аналізу точності монокулярних методів, зокрема неймережевої моноглибини. Такий підхід дозволяє забезпечити коректність експериментів і провести об'єктивний аналіз похибок у метричному просторі.

Для систематизації досліджуваних підходів та їх ключових характеристик у таблиці 3.1 наведено перелік методів вимірювання відстані, що використовуються в роботі, із зазначенням типу вхідних даних, принципу роботи та ролі оцінки глибини в кожному з підходів.

Таблиця 3.1

**Методи вимірювання відстані, що досліджуються в роботі**

<b>Метод</b>	<b>Тип даних</b>	<b>Принцип</b>	<b>Використання глибини</b>
Stereo (KITTI)	2 камери	Геометричний	Проміжне
Mono Pinhole	1 камера	Геометричний	Не використовується
Mono MDE	1 камера	Нейромережвий	Проміжне
ArUco	1 камера	Геометричний	Не використовується

У рамках експериментальних досліджень розглядаються такі ключові аспекти:

- точність вимірювання відстані до об’єктів різних класів і на різних дистанціях;
- стабільність оцінок у межах послідовності кадрів;
- вплив просторової та часової агрегації результатів на зменшення шуму;
- обчислювальна ефективність методів та їх придатність для роботи в режимі реального часу.

Для кількісної оцінки якості вимірювань використовуються стандартні метрики похибок, зокрема середня відносна похибка (AbsRel), середньоквадратична похибка (RMSE) та порогові показники точності для відстаней. Окремо аналізується вплив процедур ф’южну та згладжування на зменшення коливань оцінок відстані, що має принципове значення для практичного застосування в системах навігації АТЗ.

Таким чином, експериментальні дослідження у даній роботі спрямовані на всебічний аналіз методів вимірювання відстані, їх переваг, обмежень та областей доцільного застосування. Отримані результати створюють основу для подальшого порівняльного аналізу та формування рекомендацій щодо

вибору підходів до оцінки відстані в системах комп'ютерного зору автономних транспортних засобів.

### 3.2. Архітектура та структура програмного проєкту

Програмна реалізація експериментальної системи вимірювання відстаней до об'єктів побудована за модульним принципом, що забезпечує розділення відповідальностей між окремими компонентами та спрощує розширення й модифікацію функціональності. Такий підхід відповідає сучасним вимогам до розробки складних програмних систем комп'ютерного зору та дозволяє ізолювати етапи обробки візуальних даних, оцінки відстаней і аналізу точності результатів.

Загальна структура каталогів програмного проєкту наведена на рисунку 3.2, де відображено логічний поділ системи на модулі відповідно до етапів обробки даних.

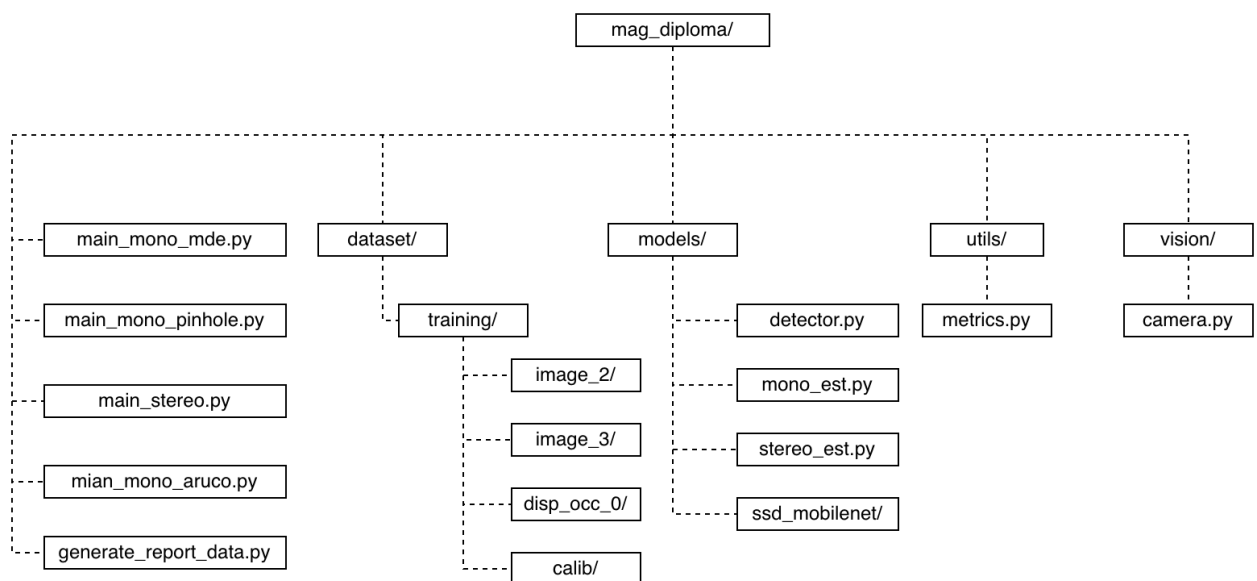


Рисунок 3.2. Структура проєкта

Каталог `datasets/` призначений для зберігання вхідних даних, які використовуються в експериментальних дослідженнях. У межах даної роботи застосовується датасет KITTI 2015, що містить стереопари зображень, карти істинної диспаритетності та калібраційні параметри камер. Наявність калібраційних файлів дозволяє виконувати перехід від диспаритету до

метричних відстаней, що є критично важливим для кількісної оцінки точності методів.

Каталог `models/` містить реалізації основних алгоритмічних компонентів системи. Зокрема, модуль `detector.py` відповідає за детекцію об'єктів на зображенні з використанням нейромережевої моделі SSD MobileNetV2. Модуль `mono_est.py` реалізує монокулярну оцінку глибини на основі моделі MiDaS, тоді `stereo_est.py` забезпечує обчислення глибини сцени за стереозображеннями. Такий поділ дозволяє незалежно використовувати та порівнювати різні підходи до оцінки відстаней у межах єдиного програмного середовища.

У каталозі `utils/` зосереджені допоміжні модулі, зокрема реалізація метрик оцінки точності. Модуль `metrics.py` містить функції для обчислення стандартних показників якості, таких як абсолютна відносна похибка (AbsRel), середньоквадратична похибка (RMSE) та показники точності за порогоми ( $\delta$ -метрики). Винесення обчислення метрик в окремий модуль забезпечує уніфікований підхід до аналізу результатів для всіх досліджуваних методів.

Каталог `vision/` призначений для роботи з джерелами зображень у режимі реального часу. Модуль `camera.py` реалізує інтерфейс потокового захоплення з камери та може бути використаний для перенесення експериментальної системи з офлайн-аналізу датасетів до сценаріїв реального застосування. Це створює основу для подальшого розширення системи в напрямі онлайн-навігації автономного транспортного засобу.

Таким чином, запропонована структура програмного проєкту відображає логіку експериментального пайплайну вимірювання відстаней — від завантаження вхідних даних і детекції об'єктів до оцінки глибини, агрегації значень у межах областей інтересу та аналізу точності результатів. Модульна організація системи забезпечує прозорість реалізації, повторюваність експериментів і зручність подальших досліджень, що є важливою вимогою для наукових робіт у галузі інженерії програмного забезпечення та комп'ютерного зору.

### 3.3. Методика вимірювання відстані до об'єктів

У межах експериментальних досліджень вимірювання відстані до об'єктів здійснюється на основі результатів обробки візуальних даних, отриманих з камер автономного транспортного засобу. На відміну від загальної оцінки глибини сцени, яка формує суцільну карту відносних або метричних значень, у даній роботі основна увага приділяється отриманню скалярної оцінки відстані до конкретного об'єкта, що має безпосереднє прикладне значення для задач навігації та безпеки руху.

Методика вимірювання відстані базується на послідовному поєднанні результатів детекції об'єктів та оцінки просторових характеристик сцени. На першому етапі за допомогою неймережевого детектора визначаються області інтересу (ROI), що відповідають об'єктам дорожньої інфраструктури, транспортним засобам або пішоходам. Кожен об'єкт описується прямокутною областю в координатах зображення, яка надалі використовується для локального аналізу просторових даних. Реалізація процесу детекції представлена у лістингу 3.1.

#### Лістинг 3.1 — Детекція об'єктів та формування областей інтересу

```
from models.detector import ObjectDetector

# Ініціалізація детектора об'єктів
detector = ObjectDetector("models/ssd_mobilenet_v2")

# Отримання детекцій з вхідного зображення
detections = detector.detect(frame, score_thr=0.5)

# Обробка кожної області інтересу
for i, det in enumerate(detections):
    ymin, xmin, ymax, xmax = det["box"]
    x1, y1 = int(xmin * W), int(ymin * H)
    x2, y2 = int(xmax * W), int(ymax * H)
    class_id = det.get("class_id", -1)
    score = det.get("score", 0.0)
```

На наступному етапі для кожної області ROI виконується агрегація значень глибини або диспаритету, отриманих відповідним методом (стерео, монокулярний підхід або калібрувальні маркери). Оскільки значення в межах ROI можуть містити шум, пропуски або локальні викиди, для отримання

максимально точної оцінки відстані використовується статистична характеристика, нечутлива до впливу викидів, зокрема медіана або квантиль розподілу значень. Такий підхід дозволяє зменшити вплив помилкових пікселів, що виникають внаслідок відблисків, слабкої текстурованості або помилок детекції. Реалізація агрегації значень глибини на основі статистичної характеристики, нечутливої до викидів, у межах області інтересу наведена у лістингу 3.2.

### Лістинг 3.2 — Агрегація значень глибини у межах області інтересу (ROI)

```
def roi_median_and_coverage(M, mask, x1, y1, x2, y2, min_valid=80):
    """
    Обчислення медіанного значення та покриття валідними пікселями
    у межах області інтересу.

    Параметри:
        M: карта глибини або диспаритету
        mask: маска валідних пікселів
        x1, y1, x2, y2: координати ROI
        min_valid: мінімальна кількість валідних пікселів

    Повертає:
        медіанне значення та коефіцієнт покриття [0..1]
    """
    if M is None:
        return float('nan'), 0.0

    H, W = M.shape[:2]
    x1, y1 = max(0, int(x1)), max(0, int(y1))
    x2, y2 = min(W - 1, int(x2)), min(H - 1, int(y2))

    if x2 <= x1 or y2 <= y1:
        return float('nan'), 0.0

    roi_M = M[y1:y2, x1:x2]
    if mask is not None:
        roi_mask = mask[y1:y2, x1:x2]
        valid_vals = roi_M[np.isfinite(roi_M) & roi_mask]
        coverage = float(np.count_nonzero(roi_mask)) / float(max(1,
roi_mask.size))
    else:
        valid_vals = roi_M[np.isfinite(roi_M)]
        coverage = float(valid_vals.size) / float(max(1, roi_M.size))

    if valid_vals.size >= min_valid:
        return float(np.median(valid_vals)), coverage
    return float('nan'), coverage
```

Отримане агреговане значення інтерпретується як оцінка відстані до відповідного об'єкта відносно камери автономного транспортного засобу. Для

стереоскопічного підходу ця відстань має безпосередній метричний зміст, тоді як для монокулярних методів вона може потребувати додаткового масштабування або вирівнювання масштабу з використанням еталонних даних. У рамках експериментів масштабування монокулярних оцінок виконується на основі статистичного вирівнювання або з використанням контрольних сцен, що дозволяє привести результати до порівнюваного метричного формату. Процедура вирівнювання масштабу монокулярної оцінки глибини відносно еталонних даних представлена у лістингу 3.3.

### Лістинг 3.3 — Вирівнювання масштабу монокулярної оцінки глибини

```
# Отримання монокулярної оцінки диспаритету
disp_pred = estimator.estimate(L)

# Вирівнювання масштабу відносно Ground Truth
if depth_gt is not None and gt_mask is not None:
    valid_pred_disp = disp_pred[gt_mask]
    valid_gt_disp = (fx * B) / (depth_gt[gt_mask] + 1e-6)

    # Медіанне масштабування
    scale = np.median(valid_gt_disp) / (np.median(valid_pred_disp) + 1e-6)

    disp_aligned = disp_pred * scale

    # Конвертація у метричну глибину
    depth_pred = (fx * B) / (disp_aligned + 1e-6)
else:
    # Без еталону використовується фіксований масштаб
    depth_pred = 100.0 / (disp_pred + 1e-6)
```

Важливою особливістю запропонованої методики є уніфікація процесу вимірювання відстані незалежно від обраного методу оцінки глибини. Використання єдиного підходу до формування ROI, агрегації значень та обчислення метрик забезпечує коректність подальшого порівняльного аналізу різних методів. Це дозволяє оцінювати не лише точність окремих алгоритмів, але й їхню практичну придатність для задач автономної навігації. Уніфікований процес обчислення відстані до об'єктів представлено у лістингу 3.4.

### Лістинг 3.4 — Уніфіковане обчислення відстані до детектованих об'єктів

```
# Адаптивний поріг валідних пікселів залежно від площі ROI
def adaptive_min_valid(x1, y1, x2, y2, cap=(20, 200), frac=0.05):
```

```

area = max(1, (x2 - x1) * (y2 - y1))
need = int(frac * area)
return max(cap[0], min(cap[1], need))

# Обчислення відстані для кожного об'єкта
for i, det in enumerate(detections):
    ymin, xmin, ymax, xmax = det["box"]
    x1, y1 = int(xmin * W), int(ymin * H)
    x2, y2 = int(xmax * W), int(ymax * H)

    mvalid = adaptive_min_valid(x1, y1, x2, y2)

    # Отримання медіанної відстані у ROI
    z_pred, coverage = roi_median_and_coverage(
        depth_pred, None, x1, y1, x2, y2, min_valid=mvalid
    )

    # Порівняння з Ground Truth (якщо доступно)
    z_gt, cov_gt = roi_median_and_coverage(
        depth_gt, gt_mask, x1, y1, x2, y2, min_valid=mvalid
    ) if depth_gt is not None else (float('nan'), 0.0)

    # Обчислення відносної похибки
    if np.isfinite(z_pred) and np.isfinite(z_gt) and cov_gt >= 0.20:
        abs_rel_error = abs(z_pred - z_gt) / (z_gt + 1e-6)

```

Таким чином, методика вимірювання відстані, застосована в роботі, орієнтована на отримання стабільних і інтерпретованих числових оцінок, які можуть безпосередньо використовуватися у модулях прийняття навігаційних рішень. Запропонований підхід створює основу для подальшого аналізу точності, стабільності та обмежень різних методів вимірювання відстані, результати якого наведено у наступних підрозділах.

### 3.4. Візуальні результати вимірювання відстані до об'єктів

Для якісної оцінки роботи розробленої системи вимірювання відстані до об'єктів у межах експериментальних досліджень було проаналізовано результати обробки реальних сцен дорожнього руху, а також дані, отримані з камери ноутбука, що використовувалися для тестування точності оцінок у внутрішніх (кімнатних) умовах. Візуальні результати дозволяють наочно продемонструвати коректність роботи алгоритмів детекції, оцінки просторових характеристик сцени та подальшого вимірювання відстані до окремих об'єктів.



У даному підрозділі наведено приклади сцен, для яких виконувалося послідовне застосування етапів обробки візуальних даних: отримання зображень з камер, детекція об'єктів, оцінка диспаритету або глибини, агрегація значень у межах областей інтересу та формування кінцевої оцінки відстані.

#### 3.4.1. Монокулярна геометрична оцінка відстані (pinhole)

Перший підхід базується на використанні геометричної моделі камери та оцінці відстані до об'єкта на основі його проєкційного розміру в зображенні. Для демонстрації роботи методу було використано як зображення з датасету KITTI (рис. 3.3).



Рисунок 3.3. Геометрична оцінка відстані до автомобіля

Отриманий результат вимірювання відстані до об'єкта за допомогою геометричної pinhole-моделі характеризується відносною похибкою на рівні 18,3%, що можна вважати задовільним показником з огляду на простоту застосованого підходу та мінімальні вимоги до вхідних даних. Даний метод ґрунтується виключно на проєкційних співвідношеннях між реальними розмірами об'єкта та його зображенням у площині кадру, без використання додаткових сенсорів або складних процедур калібрування. Крім того, на точність оцінки впливає особливості ракурсу зйомки, часткові оклюзії об'єктів та неточності локалізації області інтересу (ROI).

Додатково в межах експериментальних досліджень було проведено окремий експеримент із використанням вбудованої камери ноутбука для оцінки відстані до об'єктів у приміщенні (рис. 3.4). Метою даного експерименту є перевірка працездатності обраного підходу в умовах, відмінних від стандартних сцен датасету KITTI, а також оцінка його універсальності та можливості застосування з побутовими камерними системами без спеціалізованого калібрувального обладнання.

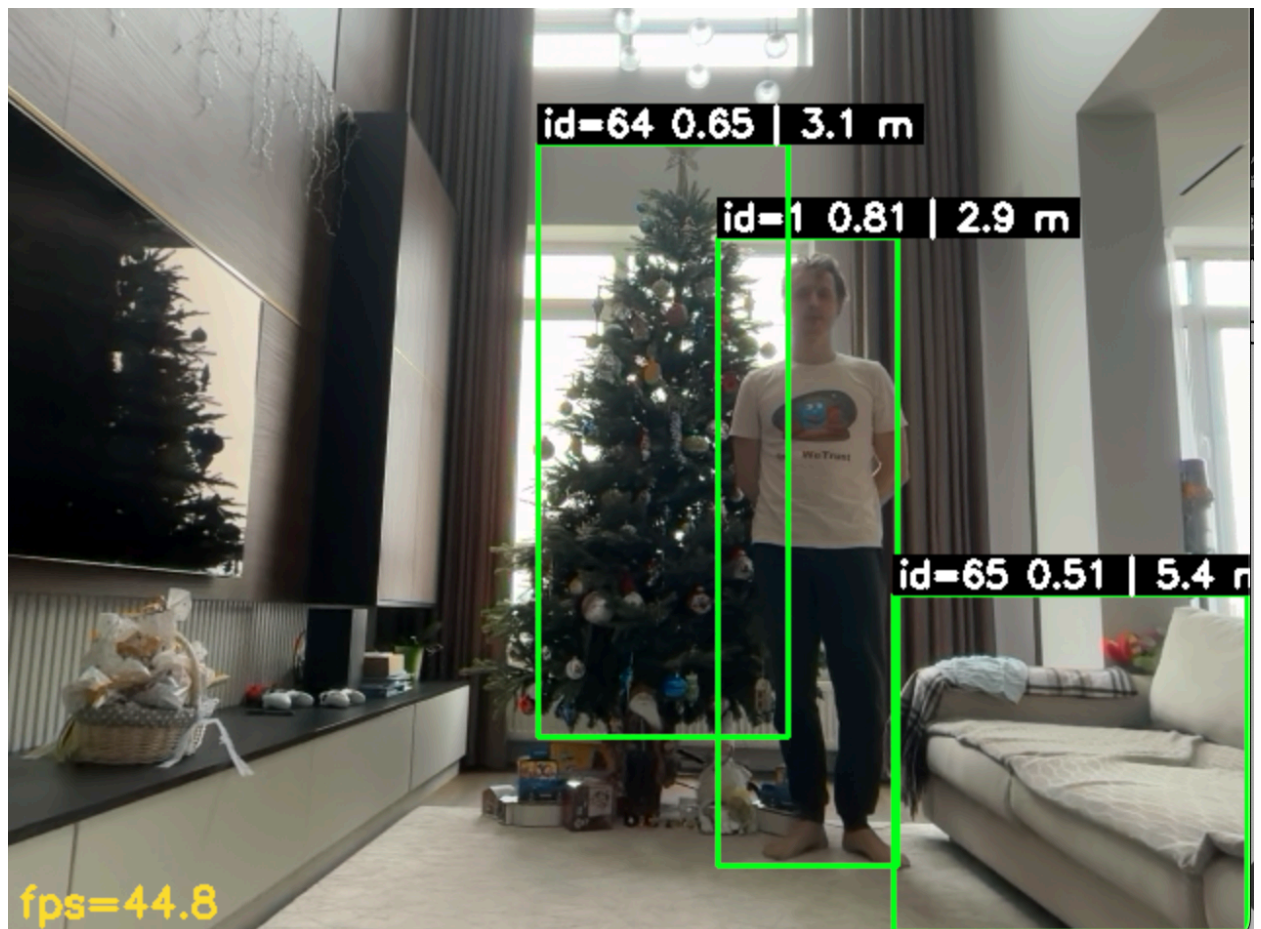


Рисунок 3.4. Геометрична оцінка відстані до людини

Отримані результати демонструють, що запропонована методика може бути адаптована для задач вимірювання відстані в контрольованих внутрішніх середовищах, де умови освітлення та геометрія сцени відрізняються від дорожніх сценаріїв. Проведений експеримент дозволяє проаналізувати вплив зміни фокусної відстані, кута огляду та роздільної здатності камери на точність геометричної оцінки, а також виявити додаткові джерела похибок, пов'язані з перспективними спотвореннями та неточністю локалізації об'єктів.



Практична значущість такого підходу полягає в можливості подальшого застосування розробленої методики для оцінки відстані до пішоходів, дорожніх знаків та інших об'єктів навігаційного середовища з використанням стандартних монокулярних камер. Це, у свою чергу, підтверджує потенціал використання геометричних методів як складової більш загальної системи вимірювання відстаней, що може бути інтегрована до програмних рішень комп'ютерного зору автономних транспортних засобів або допоміжних систем підтримки водія.

#### 3.4.2. Стереоскопічна оцінка відстані на основі датасету KITTI

Стереоскопічний підхід реалізується на основі пари синхронізованих камер та дозволяє отримати метричну оцінку глибини сцени без додаткових припущень щодо розмірів об'єктів. Для демонстрації використано дані датасету KITTI, який містить відкалібровані ліві та праві зображення.

На рисунках 3.5.-3.7. зображено реалізація роботи розробленого додатку по детекції об'єктів і оцінки відстані з використанням стерео камер:



Рисунок 3.5. Права камера

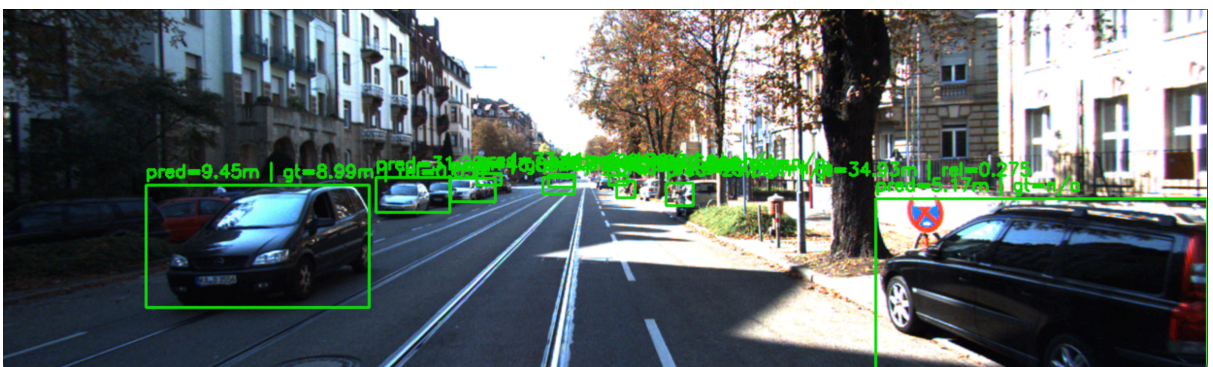


Рисунок 3.6. Ліва камера з накладанням розпізнаних об'єктів та відстані до них

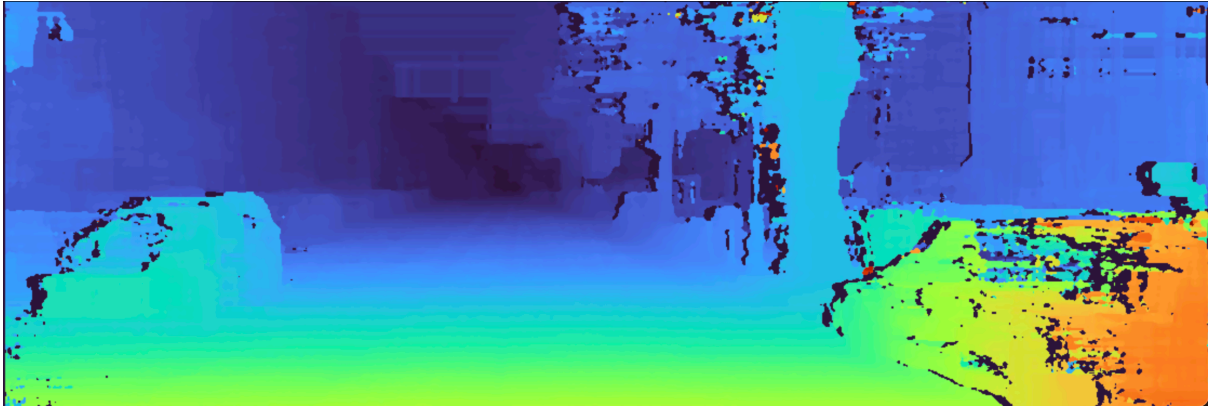


Рисунок 3.7. Диспаритет

На рисунках 3.5–3.7 представлено повний стереоскопічний ланцюг обробки: від вхідних зображень до карти диспаритету та накладання результатів вимірювання відстані. Стерео-метод забезпечує стабільні та метрично інтерпретовані оцінки відстані, однак чутливий до якості текстур сцени та наявності відблисків або однорідних поверхонь.

#### 3.4.3. Монокулярна нейромережева оцінка глибини (MDE)

Нейромережева монокулярна оцінка глибини дозволяє отримати карту глибини сцени на основі одного зображення без використання стереопари. У роботі застосовано модель MiDaS, яка формує відносну карту глибини з подальшим масштабуванням для порівняння з еталонними даними.

Для кожного виявленого об'єкта виконується агрегація значень глибини в межах bounding box. Відстань до об'єкта оцінюється за медіанним значенням глибини в області інтересу (ROI). Це дозволяє зменшити вплив локального шуму та помилкових значень (рис. 3.8-3.9).



Рисунок 3.8. Зображення з камери з накладанням оцінок відстані за допомогою MDE



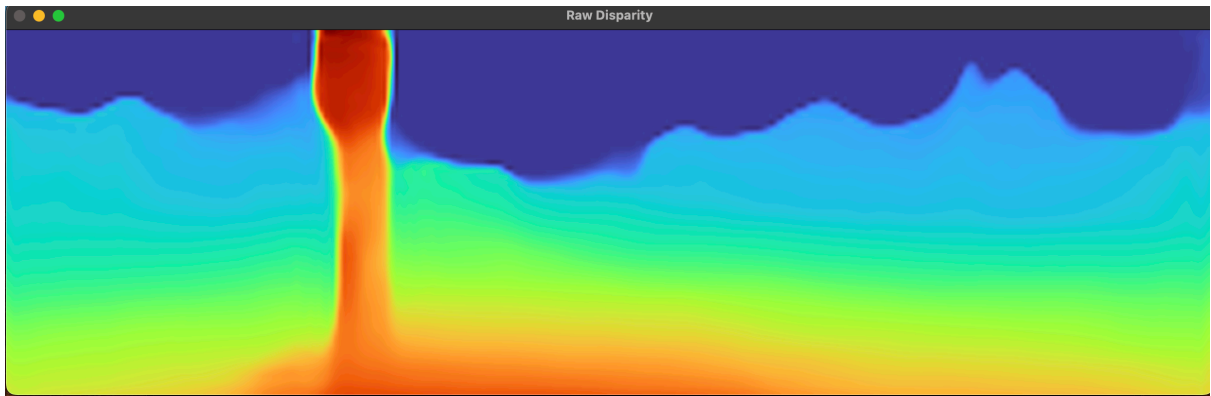


Рисунок 3.9. Сира карта диспаритету (Raw Disparity)

#### 3.4.4. Вимірювання відстані за калібрувальними маркерами ArUco

Для демонстрації еталонного підходу вимірювання відстані використано калібрувальні маркери ArUco, що дозволяють отримати метричну оцінку положення об'єкта на основі задачі визначення пози (PnP). Даний метод застосовується у контрольних або експериментальних сценах.

На рисунку 3.10 продемонстровано роботу розпізнавання ArUco міток розміщених на разній відстані до камери.

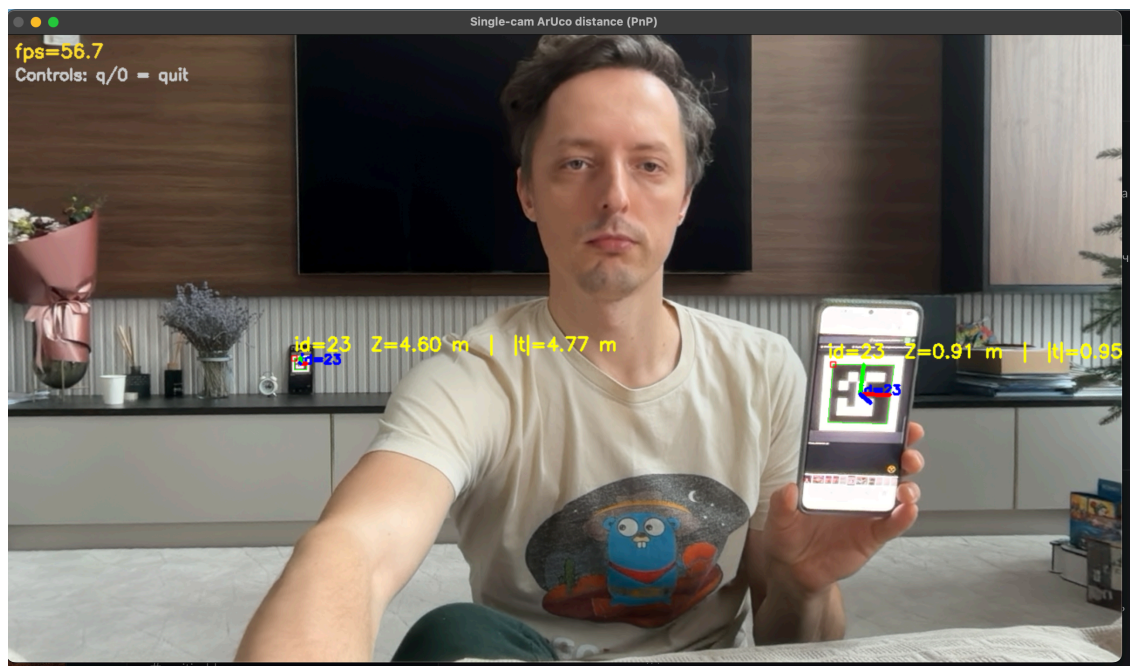


Рисунок 3.10. Результат роботи ArUco мітки

Метод ArUco забезпечив високу точність вимірювання відстані завдяки відомим геометричним параметрам маркера. Водночас його застосування обмежене спеціально підготовленими сценами, що не дозволяє використовувати його як універсальне рішення для навігації АТЗ.

### 3.5. Порівняльний аналіз методів вимірювання відстані на однакових сценах

#### 3.5.1. Повний аналіз всіх наявних сцен і об'єктів

У межах даного дослідження виконано повний цикл порівняльного аналізу трьох підходів до вимірювання відстані до об'єктів: стереоскопічного методу (Stereo), монокулярної нейромережевої оцінки глибини (Mono MDE, MiDaS) та геометричного монокулярного підходу на основі pinhole-моделі (Mono Pinhole). Аналіз проведено на повному доступному наборі даних KITTI 2015, що включає 400 сцен та 831 автоматично детектований об'єкт (транспортні засоби та пішоходи), для яких наявні еталонні дані Ground Truth.

Для наочного аналізу отриманих результатів було виконано графічну візуалізацію статистичних характеристик похибок вимірювання відстані для кожного з досліджуваних методів. Зокрема, на рисунку 3.11 наведено діаграму розподілу похибок у вигляді boxplot, яка відображає медіанні значення, міжквартильний розмах та наявність викидів для кожного підходу.

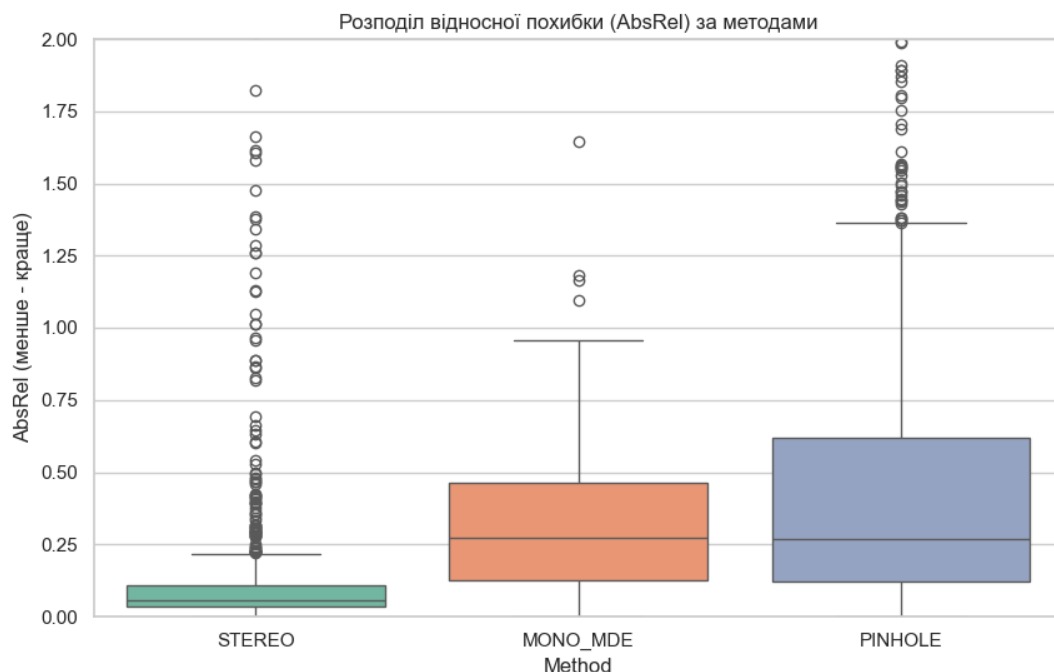


Рисунок 3.11. Діаграма розподілу похибки

Як видно з рисунка, стереоскопічний метод характеризується найменшим розмахом значень та мінімальною кількістю викидів, що свідчить

про його високу стабільність. Натомість геометричний pinhole-підхід має значну кількість аномальних значень похибки, зумовлених чутливістю методу до неточностей детекції та припущень щодо розмірів об'єктів.

Окремо приділено увагу залежності похибки вимірювання від фактичної відстані до об'єкта і проілюстровано на рисунку 3.12.

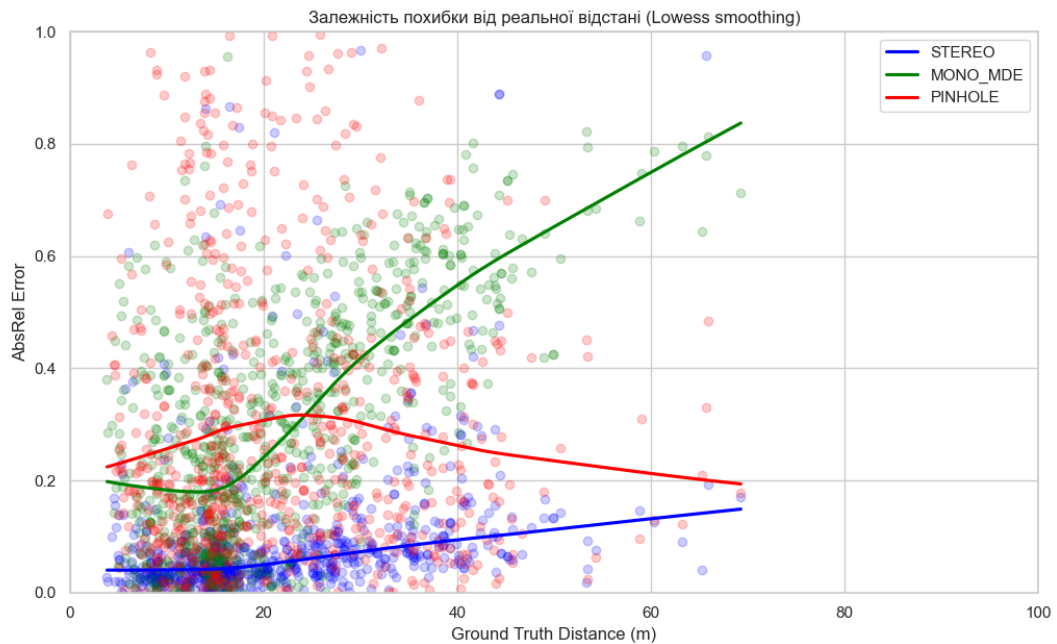


Рисунок 3.12. Графік залежності похибки від відстані

Представлений графік демонструє, що для стереоскопічного методу похибка зростає поступово зі збільшенням дистанції, що відповідає фізичним обмеженням диспаратності на великих відстанях. Для pinhole-підходу спостерігається різке зростання похибки після 40–50 м, що пояснюється зменшенням розміру об'єкта у пікселях та експоненційним накопиченням похибки геометричної проєкції. Водночас монокулярний нейромережевий метод (Mono MDE) демонструє відносно рівномірний рівень похибки в усьому діапазоні дистанцій, що є характерною властивістю моделей, які оцінюють відносну глибину сцени.

Додатково на рисунку 3.13 подано графік щільності розподілу похибок (KDE), який дозволяє оцінити концентрацію значень AbsRel для кожного методу.

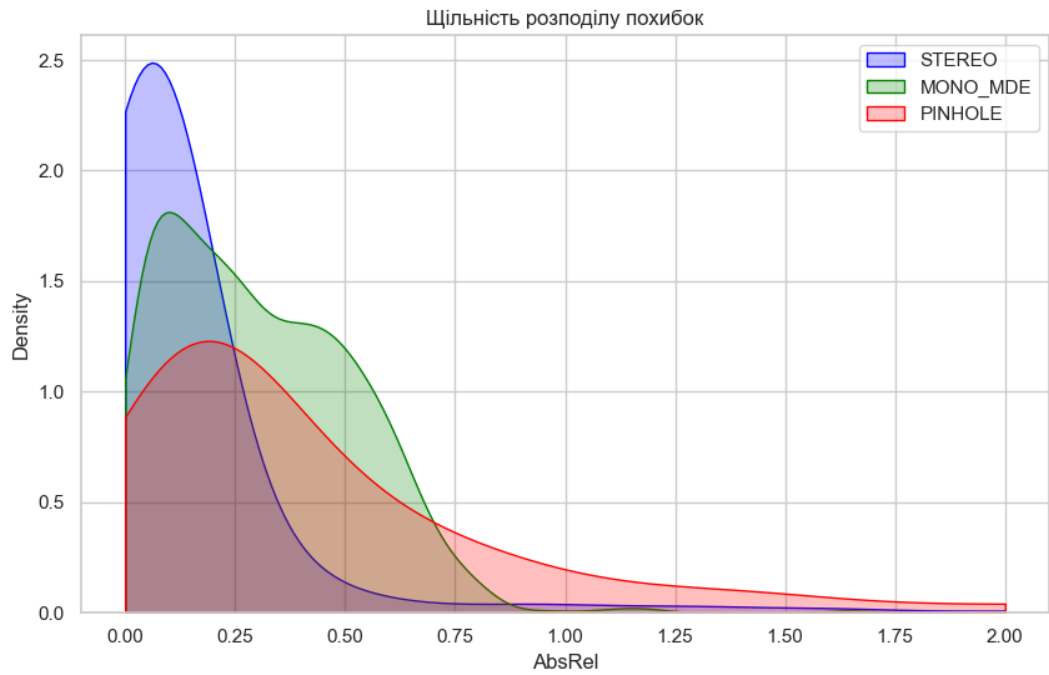


Рисунок 3.13. Графік щільності розподілу похибки

Для узагальнення отриманих результатів та спрощення їх інтерпретації основні статистичні показники точності вимірювання відстані зведено у таблицю 3.2, де наведено середні та медіанні значення відносної похибки AbsRel для кожного з досліджуваних методів.

Таблиця 3.2

**Порівняльні результати вимірювання відстані на датасеті KITTI**

Метод оцінки відстані	Кількість сцен	Кількість об'єктів	Mean AbsRel, %	Median AbsRel, %	Характеристика методу
Stereo (Disparity)	400	831	11.6	5.7	Найвища точність, стабільна метрична оцінка
Mono MDE (MiDaS)	400	831	32.6	27.3	Стабільна відносна оцінка, потребує масштабування
Mono Pinhole	400	831	51.0	26.9	Чутливий до викидів та помилок детекції



### 3.5.2. Аналіз результатів у складних умовах освітлення

Окрему групу експериментів було присвячено аналізу роботи методів вимірювання відстані в умовах поганого освітлення, зниженого контрасту та підвищеного рівня шумів. Подібні умови є типовими для реальних сценаріїв експлуатації автономних транспортних засобів і можуть суттєво впливати на якість візуальних даних.

На рисунку 3.14 наведено приклади сцен зі складними освітлювальними умовами та відповідні результати вимірювання відстані, отримані з використанням різних підходів.

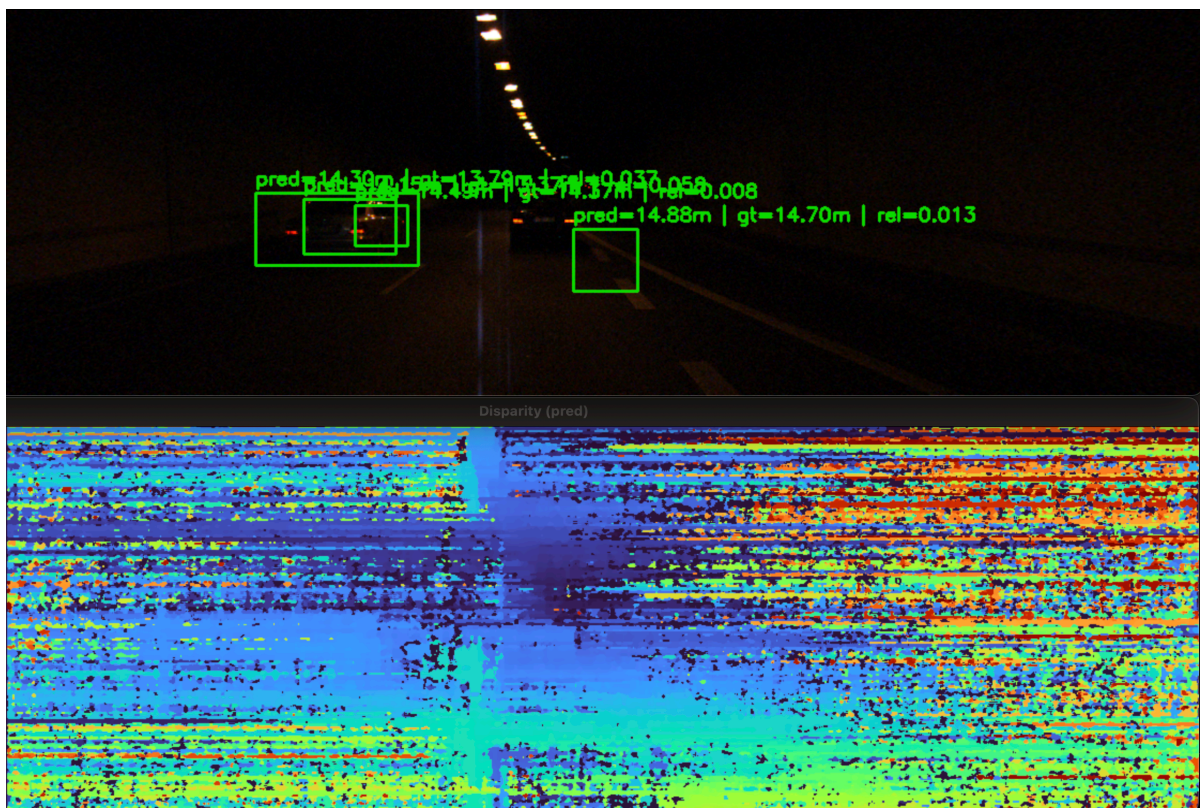


Рисунок 3.14. Шум на карті диспаритета в умовах поганого освітлення

Аналіз показує, що в умовах поганого освітлення стереоскопічні методи можуть втрачати щільність та стабільність карти диспаритету через зниження текстурованості сцени. Монокулярні нейромережеві методи в таких умовах демонструють вищу візуальну цілісність карти глибини, однак абсолютна похибка вимірювання відстані при цьому зростає. Геометричний pinhole-підхід у складних освітлювальних умовах є найбільш чутливим до помилок детекції та некоректних припущень щодо об'єктів сцени. Для наглядності

результату сцени з недостатнім освітленням, сформуємо таблицю 3.3 з результатами похибки по кожному підходу:

Таблиця 3.3

**Порівняльні результати вимірювань в умовах недостатнього освітленням**

Оцінка об'єкта	GT	Оцінка відстані	AbsRel, %
Stereo, об'єкт 1	13.37 м	14.15 м	5.8
Mono MDE, об'єкт 1	13.37 м	10.24 м	23.4
Mono Pinhole, об'єкт 1	13.37 м	20.42 м	52.8
Stereo, об'єкт 2	13.79 м	14.30 м	3.7
Mono MDE, об'єкт 2	13.79 м	10.19 м	26.1
Mono Pinhole, об'єкт 2	13.79 м	15.46 м	12.1
Stereo, об'єкт 3	14.37 м	14.49 м	0.8
Mono MDE, об'єкт 3	14.37 м	10.23 м	28.8
Mono Pinhole, об'єкт 3	14.37 м	27.7 м	93.2

Таким чином, результати експериментів у складних умовах освітлення підтверджують доцільність використання комбінованих підходів до вимірювання відстані, у яких результати окремих методів можуть взаємно доповнюватися та стабілізуватися.

*3.5.2. Аналіз ефективності ф'южну методів вимірювання відстані*

Для підвищення надійності вимірювання відстані до об'єктів у даній роботі було досліджено декілька стратегій ф'южну, що поєднують результати стереоскопічного, монокулярного нейромережевого та геометричного підходів. Метою ф'южну є зменшення впливу індивідуальних недоліків окремих методів і підвищення стабільності оцінок у складних умовах зйомки.

На рисунку 3.15 наведено порівняння розподілу відносної похибки (AbsRel) для базових методів та трьох стратегій ф'южну: простого середнього, зваженого ф'южну та стратегії з пріоритетом стереоскопічного методу. Boxplot у верхній частині рисунка 3.15 демонструє, що всі ф'южн-стратегії дозволяють суттєво зменшити кількість екстремальних викидів порівняно з монокулярними підходами, особливо з геометричним pinhole-методом. Це

свідчить про підвищення якості оцінок за рахунок комбінування джерел інформації.

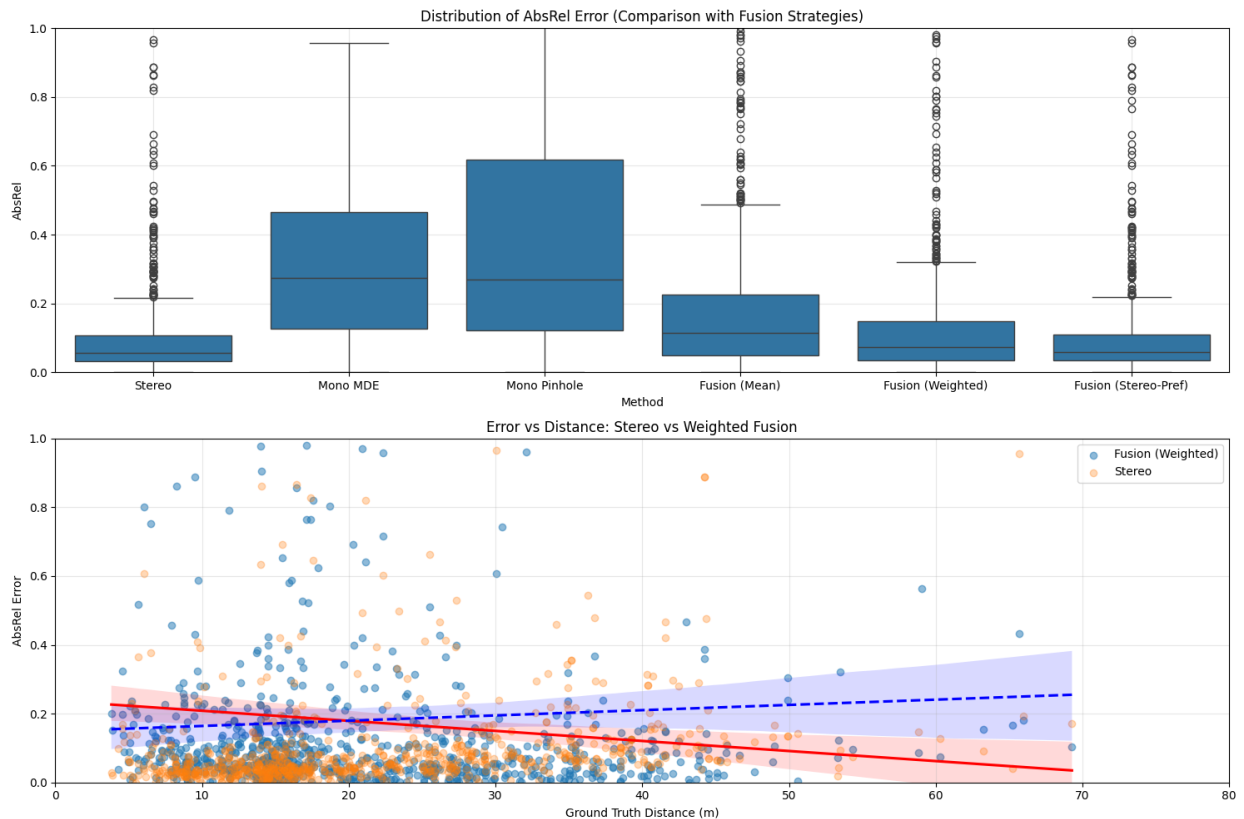


Рисунок 3.15 Графік розподілення похибки після застосування ф'южен

Водночас аналіз медіанних значень показує, що ф'южн не завжди перевершує чистий стереоскопічний метод за точністю. Стерео-оцінка характеризується найменшим міжквартильним розмахом та мінімальною медіанною похибкою, що підтверджує її високу надійність у сценаріях, де доступні якісні стереодані. Ф'южн-стратегії, хоча й покращують стабільність у порівнянні з mono MDE та pinhole, демонструють дещо вищу медіанну похибку, що пояснюється внеском менш точних компонентів.

Нижня частина рисунка 3.15 ілюструє залежність похибки від реальної відстані до об'єкта для стереоскопічного методу та зваженого ф'южну. Як видно з графіка, на малих і середніх дистанціях (до приблизно 30–40 м) зважений ф'южн демонструє результати, порівнянні зі stereo, а в окремих випадках — меншу дисперсію похибок. Це пояснюється тим, що в цій зоні всі

методи формують відносно узгоджені оцінки, а ф'южн дозволяє частково компенсувати локальні помилки окремих алгоритмів.

Однак із зростанням відстані ефективність ф'южну поступово знижується. На великих дистанціях ( $>40\text{--}50\text{ м}$ ) похибка монокулярних методів (MDE та pinhole) зростає швидше, ніж у стереоскопічного підходу, що призводить до деградації зваженого ф'южну. У таких умовах додавання менш точних оцінок починає негативно впливати на результат, і чистий stereo-метод забезпечує кращу точність, що чітко простежується за трендами на рисунку 3.15.

Отримані результати дозволяють зробити важливий висновок: ф'южн методів вимірювання відстані не є універсальним засобом підвищення точності, а його ефективність істотно залежить від умов сцени та діапазону відстаней. У випадках, коли доступні якісні стереодані, використання стереоскопічного методу як основного є більш доцільним. Водночас ф'южн виправданий у сценаріях часткової втрати даних, поганої текстурованості сцени або нестабільної роботи окремих модулів, де він дозволяє підвищити стійкість системи та зменшити кількість грубих помилок.

### **3.6. Аналіз продуктивності та часових характеристик підходів**

Оцінка продуктивності є критично важливою складовою аналізу систем вимірювання відстані в автономних транспортних засобах, оскільки всі розглянуті алгоритми мають працювати в умовах обмеженого часу обробки та наближеного до реального часу режиму. У межах даної роботи було проведено експериментальний аналіз часових характеристик основних компонентів системи для трьох підходів: стереоскопічного, геометричного pinhole та монокулярного нейромережевого (MDE, MiDaS).

Вимірювання продуктивності виконувалися на обчислювальній платформі на базі процесора Apple M-серії (або еквівалентній за продуктивністю конфігурації), без використання спеціалізованих апаратних прискорювачів (GPU або NPU). Для кожного підходу оцінювався середній час обробки одного кадру з урахуванням усіх етапів обчислювального пайплайну:

отримання зображень, детекції об'єктів, оцінки просторових характеристик, а також візуалізації результатів.

У таблиці 3.4 наведено узагальнені результати вимірювання часової складності обробки одного кадру для кожного з досліджуваних методів.

*Таблиця 3.4*

**Часова складність обробки одного кадру**

Модуль	Стерео	Pinhole	MDE (MiDaS)
Отримання кадрів	5 мс	2.5 мс	2.5 мс
Детекція об'єктів	106 мс	106 мс	15–600 мс
Обчислення глибини	12 мс	~1 мс	~470 мс
Візуалізація	3–4 мс	2–3 мс	2–3 мс
Загальна затримка	~126 мс	~112 мс	>600 мс
Досяжний FPS	~8	~9	<2

Як видно з таблиці 3.4, основним вузьким місцем системи є нейромережевий модуль детекції об'єктів, який забезпечує найбільший внесок у загальну затримку незалежно від обраного методу вимірювання відстані. Для стереоскопічного та pinhole-підходів час детекції є стабільним і становить близько 100–110 мс на кадр, що визначає верхню межу досяжної частоти кадрів.

Додаткові обчислення, пов'язані зі стереоскопічною оцінкою глибини, збільшують загальну затримку системи лише приблизно на 10% порівняно з геометричним pinhole-методом. При цьому стерео-підхід забезпечує значно вищу точність вимірювання відстані, що робить таке збільшення обчислювальних витрат виправданим з практичної точки зору. Отримані результати свідчать, що стереоскопічний метод є компромісним рішенням між точністю та продуктивністю для задач автономної навігації.

Натомість монокулярний нейромережевий підхід (MDE, MiDaS) характеризується істотно вищими обчислювальними витратами. Час оцінки глибини для одного кадру в CPU-режимі перевищує 450 мс, а загальна затримка системи перевищує 600 мс, що обмежує досяжну частоту обробки менш ніж 2 кадрами за секунду. Крім того, значна варіативність часу детекції

та інференсу мережі (15–600 мс) ускладнює забезпечення стабільної роботи системи в реальному часі.

Таким чином, результати аналізу продуктивності показують, що геометричний pinhole-метод та стереоскопічний підхід є придатними для роботи в умовах обмеженого часу обробки, тоді як застосування монокулярної нейромережевої оцінки глибини без апаратного прискорення є недоцільним для реальних систем автономного керування. У практичних застосуваннях MDE доцільно розглядати як допоміжний або офлайн-інструмент, або використовувати його лише за наявності GPU/NPU-прискорення.

### **Висновки до розділу 3**

У третьому розділі роботи було проведено комплексні експериментальні дослідження методів вимірювання відстані до об'єктів у системах комп'ютерного зору автономних транспортних засобів. На відміну від абстрактної оцінки глибини сцени, основна увага була зосереджена на отриманні скалярних метричних оцінок відстані до конкретних об'єктів дорожньої інфраструктури, що має безпосереднє прикладне значення для задач навігації та безпеки руху.

У межах експериментів було реалізовано та досліджено три підходи: стереоскопічний метод, монокулярну нейромережеву оцінку глибини (MDE, MiDaS) та геометричний pinhole-підхід. Для забезпечення коректного порівняльного аналізу всі методи були інтегровані в уніфікований обчислювальний пайплайн, який включає детекцію об'єктів, формування областей інтересу (ROI), агрегацію просторових даних та обчислення відносних і абсолютних похибок.

Експериментальні дослідження на датасеті KITTI показали, що стереоскопічний підхід забезпечує найвищу точність і стабільність вимірювання відстані, з медіанною відносною похибкою на рівні 5–6%. Висока надійність цього методу зумовлена прямим використанням геометричних співвідношень між двома камерами та наявністю метричного

масштабу, що робить його найбільш придатним для задач автономної навігації.

Монокулярний нейромережевий підхід (MDE) продемонстрував здатність до стабільної оцінки відносних відстаней незалежно від дальності об'єктів, однак поступився геометричним методам за абсолютною метричною точністю. Навіть після вирівнювання масштабу середні та медіанні значення похибки залишаються суттєвими, що обмежує можливість використання MDE як єдиного джерела інформації про відстань у критичних навігаційних сценаріях.

Геометричний pinhole-метод показав найвищу варіативність результатів. Незважаючи на прийнятні медіанні значення похибки, його середня похибка є значною через чутливість до якості детекції об'єктів та припущень щодо їх реальних розмірів. Це підтверджує, що pinhole-підхід може застосовуватися лише як допоміжний або резервний метод у контрольованих умовах.

Окрему увагу було приділено аналізу ф'южну результатів різних методів. Експериментальні дані показали, що ф'южн дозволяє зменшити похибку на малих та середніх дистанціях, а також підвищити стійкість системи у випадках часткової деградації окремих методів. Водночас на великих відстанях якість ф'южну обмежується накопиченням похибок монокулярних методів, що призводить до погіршення результатів порівняно з чистим стереоскопічним підходом.

Аналіз продуктивності показав, що стереоскопічний та pinhole-підходи є придатними для роботи в наближеному до реального часу режимі, тоді як монокулярні нейромережеві методи без апаратного прискорення мають істотні обмеження за швидкодією. Це підтверджує необхідність врахування не лише точності, але й обчислювальних витрат при виборі методів для практичного використання в автономних системах.

Таким чином, результати третього розділу підтверджують доцільність використання гібридного підходу, у якому стереоскопічний метод виступає

основним джерелом метричних оцінок відстані, а монокулярні та геометричні методи — допоміжними компонентами для підвищення стійкості та покриття системи. Отримані експериментальні результати створюють основу для формування загальних висновків роботи та визначення напрямів подальших досліджень.



## ВИСНОВКИ

У магістерській роботі розв’язано актуальну науково-практичну задачу вимірювання відстані до об’єктів у системах комп’ютерного зору автономних транспортних засобів, що має ключове значення для забезпечення безпеки навігації та прийняття рішень у реальному часі. Особливу увагу приділено інженерним аспектам реалізації програмних рішень.

У першому розділі виконано ґрунтовний аналіз сучасних підходів до побудови систем комп’ютерного зору для автономних транспортних засобів. Розглянуто геометричні та нейромережеві методи обробки зображень, моделі камери, принципи оцінки глибини сцени та просторових параметрів об’єктів. Показано, що для практичних задач автономного керування ключовим є не формування суцільної карти глибини, а отримання надійних метричних оцінок відстані до конкретних об’єктів, які безпосередньо використовуються у навігаційних модулях.

У другому розділі обґрунтовано вибір програмних інструментів та бібліотек для реалізації досліджуваної системи. Запропоновано модульну архітектуру програмного забезпечення, що дозволяє поєднувати класичні алгоритми комп’ютерного зору з методами глибокого навчання. Використання OpenCV, TensorFlow та відкритого датасету KITTI забезпечило відтворюваність експериментів, масштабованість програмного рішення та можливість його адаптації до різних апаратних конфігурацій.

У третьому розділі проведено комплексні експериментальні дослідження методів вимірювання відстані до об’єктів із використанням стереоскопічного підходу, монокулярної нейромережевої оцінки глибини (MDE) та геометричного pinhole-методу. Запропоновано уніфіковану методику обробки результатів, що включає детекцію об’єктів, формування областей інтересу, агрегацію просторових даних і оцінку похибок. Результати експериментів підтвердили, що стереоскопічний підхід забезпечує найвищу точність і стабільність, тоді як монокулярні методи можуть бути ефективно використані за умови відповідного масштабування та фіксовану результатів.

Важливим практичним результатом роботи є підтвердження того, що запропонований програмний підхід є працездатним не лише у спеціалізованих експериментальних середовищах, а й у звичайних умовах експлуатації. Проведені додаткові експерименти з використанням вбудованої камери ноутбука показали можливість коректної оцінки відстаней у приміщенні без застосування дороговартісних сенсорів або спеціалізованих камер. Це свідчить про потенціал розробленої системи для використання у доступних та бюджетних програмно-апаратних рішеннях, зокрема у навчальних, допоміжних та напівіндустріальних системах комп'ютерного зору.

Аналіз продуктивності показав, що геометричні та стереоскопічні методи можуть застосовуватися у режимі, близькому до реального часу, без значних апаратних вимог, тоді як нейромережеві підходи потребують апаратного прискорення для повноцінного використання у динамічних сценаріях. Це підтверджує доцільність інженерного підходу до вибору алгоритмів із урахуванням обчислювальних ресурсів та вартості системи.

Таким чином, результати магістерської роботи мають практичну цінність для інженерії програмного забезпечення, оскільки демонструють можливість створення гнучких, масштабованих і економічно доцільних систем комп'ютерного зору для вимірювання відстані до об'єктів. Запропоновані рішення можуть бути використані як основа для подальших досліджень, розширення багатомодального ф'южну, інтеграції апаратного прискорення та розробки доступних автономних або напівавтономних систем, що не потребують дорогих сенсорних комплексів.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Козуб Г. О., Козуб Ю. Г. Декларативний підхід при створенні мультиплатформних додатків. *Вісник Східноукраїнського національного університету імені Володимира Даля*, 2022. №5 (275). С. 10–15. DOI: <https://doi.org/10.33216/1998-7927-2022-275-5-10-15>
2. Козуб Г., Крутько О.О. Алгоритми компресії зображень без втрати інформації. *Proceedings of the XIII International Scientific and Practical Conference*. Edmonton, Canada. 2023. Pp. 510-513. URL: <https://isg-konf.com/information-activity-as-a-component-of-science-development/> Available at: DOI: 10.46299/ISG.2023.1.13
3. Abduvalieva G, Barsanaeva D, Kenenbaeva G, Kozub H, Aghayeva S. Innovations in educational methodologies: Exploring the impact of digital technologies on learning and teaching. *Sci Herald Uzhhorod Univ Ser Phys*. 2024;(55):2890-2899. DOI: 10.54919/physics/55.2024.289q10. <https://physics.uz.ua/en/article/503e3b6270e06a95843cd70f583801e8> Scopus
4. Козуб В.Ю., Козуб Г.О. Використання алгоритмів для аналізу та обробки зображень у різних сферах, таких як медицина та безпека. «Таврійський науковий вісник. Серія: Технічні науки», Херсон : Видавничий дім «Гельветика», 2025. (1), 24-32. вилучено із <https://journals.ksauniv.ks.ua/index.php/tech/article/view/777> DOI <https://doi.org/10.32782/tnv-tech.2025.1.3>
5. Лень О.Ю., Козуб Г.О. Сучасні підходи до роботи з комп'ютерною графікою та геометричним моделюванням у САПР. *Global trends in science and education. Proceedings of the 5th International scientific and practical conference*. SPC “Sci-conf.com.ua”. Kyiv, Ukraine. 2025. Pp. 376-378. URL: <https://sci-conf.com.ua/v-mizhnarodna-naukovo-praktichna-konferentsiya-global-trends-in-science-and-education-2-4-06-2025-kiyiv-ukrayina-arhiv/>
6. Козуб Г.О., Крамар А. В., Базалійська О.В. Використання GPU для паралельної обробки графіки та відео в мобільних додатках. *Наука і техніка*

сьогодні. № 7(48) 2025. С. 1607-1620. [https://doi.org/10.52058/2786-6025-2025-7\(48\)-1607-1620](https://doi.org/10.52058/2786-6025-2025-7(48)-1607-1620)

7. Руденко О., Козуб Г. Застосування методів глибокого навчання в програмних рішеннях комп'ютерного зору для автономних систем. *Current scientific goals, approaches and challenges: collection of scientific papers «SCIENTIA» with Proceedings of the IV International Scientific and Theoretical Conference, June 13, 2025*. Dresden, Federal Republic of Germany: International Center of Scientific Research. 2025. Pp.150-153. <https://previous.scientia.report/index.php/archive/issue/view/13.06.2025>
8. Alhashim A., Li Y. Monocular Depth Estimation Using Deep Learning: A Review // *Sensors*. 2022. DOI: <https://doi.org/10.3390/s22145353>.
9. Wang Y., Zhang J., Liu Z. et al. Deep Learning-based Depth Estimation Methods from Monocular Image and Videos: A Comprehensive Survey // *ACM Computing Surveys*. 2024. DOI: <https://doi.org/10.1145/3677327>.
10. Alhashim A., Li Y. Deep Learning-Based Monocular Depth Estimation Methods - A State-of-the-Art Review // *Sensors*. 2020. DOI: <https://doi.org/10.3390/s20082272>.
11. Zhang Y. et al. Monocular Depth Estimation Based on Deep Learning: An Overview. 2020. arXiv:2003.06620. DOI: <https://doi.org/10.48550/arXiv.2003.06620>.
12. Li X. et al. Deep Learning for Monocular Depth Estimation: A Review // *Neurocomputing*. 2021. DOI: <https://doi.org/10.1016/j.neucom.2020.12.013>.
13. Singh A. et al. Object Detection for Autonomous Vehicle Using TensorFlow // *Lecture Notes in Computer Science*. 2019. DOI: [https://doi.org/10.1007/978-3-030-30465-2\\_11](https://doi.org/10.1007/978-3-030-30465-2_11).
14. Sharma S. et al. Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions. 2023. arXiv:2311.09093. DOI: <https://doi.org/10.48550/arXiv.2311.09093>.

15. Kim J. et al. Autonomous Driving in Traffic with End-to-End Vision-Based Deep Learning // Neurocomputing. 2024. DOI: <https://doi.org/10.1016/j.neucom.2024.127645>.
16. Patel R. et al. Real-Time Object Detection in Autonomous Vehicles with YOLO // World Electric Vehicle Journal. 2024. DOI: <https://doi.org/10.3390/wevj16010009>.
17. Lee S. et al. Estimating Traveled Distance from Monocular Images Using a Recurrent Convolutional Neural Network. 2019. arXiv:1904.08105. DOI: <https://doi.org/10.48550/arXiv.1904.08105>.
18. Ivanov D. et al. Accurate 3D to 2D Object Distance Estimation from the Mapped Point Cloud Data // Sensors. 2023. DOI: <https://doi.org/10.3390/s23042103>.
19. Chen H. et al. Absolute Distance Prediction Based on Deep Learning from Monocular Camera. 2021. arXiv:2111.01715. URL: <https://arxiv.org/pdf/2111.01715>.
20. Park J. et al. A Convolutional Neural-Network-Based Training Model to Estimate Actual Distance of Persons in Continuous Images // Sensors. 2022. DOI: <https://doi.org/10.3390/s22155811>.
21. Abadi M. et al. TensorFlow: A System for Large-Scale Machine Learning. 2016. arXiv:1605.08695. DOI: <https://doi.org/10.48550/arXiv.1605.08695>.
22. Ranftl R., Bochkovskiy A., Koltun V. Vision Transformers for Dense Prediction. 2021. arXiv:2103.13413. DOI: <https://doi.org/10.48550/arXiv.2103.13413>.
23. Geiger A., Lenz P., Urtasun R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2012. DOI: <https://doi.org/10.1109/CVPR.2012.6248074>.

24. Hirschmüller H. Stereo Processing by Semi-Global Matching and Mutual Information // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2008. DOI: <https://doi.org/10.1109/TPAMI.2007.1166>.
25. Bradski G. The OpenCV Library // Dr. Dobb's Journal of Software Tools. 2000.
26. Zhang Z. A Flexible New Technique for Camera Calibration // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000. DOI: <https://doi.org/10.1109/34.888718>.
27. Garrido-Jurado S., Muñoz-Salinas R., Madrid-Cuevas F. J., Medina-Carnicer R. Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion // Pattern Recognition. 2014. DOI: <https://doi.org/10.1016/j.patcog.2014.01.005>.
28. Lowe D. G. Distinctive Image Features from Scale-Invariant Keypoints // International Journal of Computer Vision. 2004. DOI: <https://doi.org/10.1023/B:VISI.0000029664.99615.94>.
29. Kalman R. E. A New Approach to Linear Filtering and Prediction Problems // Journal of Basic Engineering. 1960. DOI: <https://doi.org/10.1115/1.3662552>.
30. Bewley A., Ge Z., Ott L., Ramos F., Upcroft B. Simple Online and Realtime Tracking (SORT) // Proceedings of the IEEE International Conference on Image Processing (ICIP). 2016. DOI: <https://doi.org/10.1109/ICIP.2016.7533003>.
31. Ranftl R., Bochkovskiy A., Koltun V. Vision Transformers for Dense Prediction // Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2021. DOI: <https://doi.org/10.48550/arXiv.2103.13413>.
32. Chang J.-R., Chen Y.-S. Pyramid Stereo Matching Network // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2020. DOI: <https://doi.org/10.1109/CVPR42600.2020.00063>.
33. Tosi F., Aleotti F., Poggi M., Mattoccia S. Learning Monocular Depth Estimation Infusing Traditional Stereo Knowledge // Proceedings of the IEEE

Conference on Computer Vision and Pattern Recognition (CVPR). 2019. DOI: <https://doi.org/10.1109/CVPR.2019.00101>.

34. Godard C., Mac Aodha O., Firman M., Brostow G. J. Digging into Self-Supervised Monocular Depth Estimation // Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2019. DOI: <https://doi.org/10.1109/ICCV.2019.00383>.

35. Bhat S. F., Alhashim I., Wonka P. AdaBins: Depth Estimation Using Adaptive Bins // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2021. DOI: <https://doi.org/10.1109/CVPR46437.2021.00384>.

36. Liu S., Zhang J., Zhang S., Metaxas D. Monocular Depth Estimation via Learned Scale Recovery // Proceedings of the European Conference on Computer Vision (ECCV). 2022. URL: <https://arxiv.org/pdf/2003.06620>.

37. Sakaridis C., Dai D., Van Gool L. Guided Curriculum Model Adaptation and Uncertainty-Aware Evaluation for Semantic Nighttime Image Segmentation // Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2021. DOI: <https://doi.org/10.1109/ICCV48922.2021.00356>.

# ДОДАТОК А

## Сертифікат апробації і впровадження





## ДОДАТОК Б. ВИХІДНИЙ КОД ДОДАТКУ

### Скрипт з використанням монокулярної геометрії pinhole

```
import os
import time
from typing import Dict, Optional, Tuple

import cv2
import numpy as np

from models.detector import ObjectDetector
from vision.camera import CameraStream

# ----- Config -----
# Якщо є калібрування камери (інтрінсики) – покладіть файл з масивами K, dist
# у форматі npz: {"K": 3x3, "dist": (k1,k2,p1,p2[,k3])}
INTR_FILE = "camera_intrinsics.npz"

# Якщо інтрінсіків немає – будемо оцінювати fx за кутом огляду
ASSUME_HFOV_DEG = 60.0

# Підказки реальних розмірів по класах (COCO id -> H_real_approx у метрах)
# Орієнтовні значення: людина ~1.70 м; автомобіль ~1.50 м; велосипед ~1.40 м
# (з людиною)
COCO_HEIGHT_HINTS: Dict[int, float] = {
    1: 1.70, # person
    2: 1.40, # bicycle (з людиною)
    3: 1.50, # car
    4: 1.60, # motorcycle (з людиною)
    6: 2.30, # bus (видима висота)
    8: 3.20, # truck (середня видима)
}

# Мінімальна висота bbox у пікселях, нижче якої оцінка нестабільна
MIN_BBOX_H_PX = 20

# ----- Utils -----
def load_intrinsics_or_guess(w: int, h: int) -> Tuple[np.ndarray,
np.ndarray]:
    """
    Повертає (K, dist). Якщо немає файлу intrinsics – оцінюємо fx із
    припущення HFOV.
    """
    if os.path.exists(INTR_FILE):
        data = np.load(INTR_FILE)
        K = data["K"].astype(np.float32)
        dist = data.get("dist", np.zeros(5, np.float32)).astype(np.float32)
        print(f"[calib] loaded intrinsics from {INTR_FILE}: fx={K[0,0]:.1f},
fy={K[1,1]:.1f}")
        return K, dist
```

```

# Approximate intrinsics from HFOV
fx = 0.5 * w / np.tan(np.deg2rad(ASSUME_HFOV_DEG) * 0.5)
fy = fx
cx, cy = w / 2.0, h / 2.0
K = np.array([[fx, 0, cx], [0, fy, cy], [0, 0, 1]], dtype=np.float32)
dist = np.zeros(5, np.float32)
print(f"[calib] using approximate intrinsics (HFOV≈{ASSUME_HFOV_DEG}°):
fx={fx:.1f}")
return K, dist

def pinhole_distance_from_bbox(
    box_xyxy: Tuple[int, int, int, int],
    cls_id: int,
    fx: float,
    height_hints: Dict[int, float],
    default_h: float = 1.70,
) -> Optional[float]:
    """
    Оцінка дистанції за pinhole-моделлю:  $Z \approx fx * H_{real} / h_{img}$ .
    Використовуємо приблизну реальну висоту об'єкта (за класом COCO).
    """
    x1, y1, x2, y2 = map(int, box_xyxy)
    h_px = max(0, y2 - y1)
    if h_px < MIN_BBOX_H_PX or fx <= 0:
        return None
    H_real = float(height_hints.get(cls_id, default_h))
    Z = (fx * H_real) / float(h_px)
    if not np.isfinite(Z) or Z <= 0:
        return None
    return float(Z)

def draw_box_with_label(img: np.ndarray, xyxy: Tuple[int, int, int, int],
    label: str, color=(36, 255, 12)):
    x1, y1, x2, y2 = map(int, xyxy)
    cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)
    # тло для тексту
    (tw, th), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 2)
    y0 = max(0, y1 - th - 6)
    cv2.rectangle(img, (x1, y0), (x1 + tw + 6, y0 + th + 6), (0, 0, 0), -1)
    cv2.putText(img, label, (x1 + 3, y0 + th + 2), cv2.FONT_HERSHEY_SIMPLEX,
    0.6, (255, 255, 255), 2)

# ----- Main loop -----
def main():
    print("🚗 Starting mono distance (pinhole) - main_mono2.py")

    cam = CameraStream(0)

    # Отримаємо один кадр (warm-up), щоб оцінити K
    frame = cam.read()
    if frame is None:

```

```

# короткий warm-up на macOS/AVFoundation може бути потрібним
t0 = time.time()
while frame is None and (time.time() - t0) < 2.0:
    time.sleep(0.01)
    frame = cam.read()
if frame is None:
    cam.release()
    raise RuntimeError("[camera] no frames from camera (index=0). Ensure
permissions and that the camera is not busy.")
h, w = frame.shape[:2]
K, _ = load_intrinsics_or_guess(w, h)
fx = float(K[0, 0])

detector = ObjectDetector("models/ssd_mobilenet_v2")

fps_ema = None
while True:
    t0 = time.time()
    frame = cam.read()
    if frame is None:
        # тимчасовий збій джерела – спробуємо продовжити
        time.sleep(0.01)
        continue

    # Детекція: коробки у нормалізованих координатах [ymin, xmin, ymax,
xmax]
    detections = detector.detect(frame)

    vis = frame.copy()
    if detections:
        for d in detections:
            # Розпакувати bbox
            ymin, xmin, ymax, xmax = d["box"]
            x1 = int(xmin * w)
            y1 = int(ymin * h)
            x2 = int(xmax * w)
            y2 = int(ymax * h)
            cls_id = int(d.get("class_id", -1))
            score = float(d.get("score", 0.0))

            Z = pinhole_distance_from_bbox((x1, y1, x2, y2), cls_id, fx,
COCO_HEIGHT_HINTS)
            if Z is not None:
                label = f"id={cls_id} {score:.2f} | {Z:.1f} m"
            else:
                label = f"id={cls_id} {score:.2f}"
            draw_box_with_label(vis, (x1, y1, x2, y2), label)
        else:
            cv2.putText(vis, "No detections", (10, 28),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 200, 255), 2)

    # FPS
    dt = time.time() - t0
    fps_ema = (0.9 * fps_ema + 0.1 * (1.0 / dt)) if fps_ema else (1.0 /
dt)

```

```

        cv2.putText(vis, f"fps={fps_ema:.1f}", (10, h - 12),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (50, 220, 255), 2)

        cv2.imshow("Mono pinhole distance (main_mono2)", vis)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cam.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()

```

## Скрипт бінокулярної оцінки відстані за диспаритетом з використанням датасету KITTI

```
# main_kitti_eval.py
import os
import glob
import csv
import cv2
import numpy as np

import time
from models.depth_estimator_stereo import StereoDepthEstimator, StereoCalib
from models.detector import ObjectDetector
from datasets.kitti_loader import load_kitti_pair
from utils.metrics import compute_depth_metrics

# === Налаштування ===
KITTI_ROOT = "datasets/kitti2015/training" # шлях до training/
SCORE_THR = 0.5
GT_COVERAGE_THR = 0.20 # мін. частка валідних GT пікселів у ROI (0..1)
# для метрики
MIN_MEDIAN_DISP = 2.0 # мін. медіанна disparity в ROI (px), щоб не брати
# «наддалекі»
CLICK_HALF = 30 # півсторона ROI при кліку (60x60)
CSV_PATH_PER_FRAME = "kitti_eval_frame.csv" # зберегти поточний кадр
# (клавіша 'w')
CSV_PATH_ALL = "kitti_eval_all.csv" # акумуляція по всіх кадрах
# (клавіша 'W')

# Параметри візуалізації/кліпу глибини
DEPTH_CLIP = (0.2, 200.0)

# === Хелпери ===
def load_gt_disparity_png16(path_png):
    """GT disparity PNG16 зі шкалою 256 → float disparity (px), nan там, де
    даних немає."""
    if path_png is None or not os.path.exists(path_png):
        return None
    disp_png = cv2.imread(path_png, cv2.IMREAD_UNCHANGED)
    if disp_png is None:
        return None
    d = disp_png.astype(np.float32) / 256.0
    d[d <= 0] = np.nan
    return d

def depth_from_disp(d, fx, B, clip=DEPTH_CLIP):
    """ $Z$  (м) =  $fx * B / d$ , з обрізанням для стабільного відображення."""
    Z = (fx * B) / d
    if clip:
        Z = np.clip(Z, clip[0], clip[1])
    return Z

def valid_mask_from_disparity(d):
    """Маска валідних GT-пікселів з disparity (True там, де GT є)."""
```

```

if d is None:
    return None
return np.isfinite(d) & (d > 0)

def adaptive_min_valid(x1, y1, x2, y2, cap=(20, 200), frac=0.05):
    """Адаптивний поріг валідних пікселів: 5% площі ROI, в межах cap."""
    area = max(1, (x2 - x1) * (y2 - y1))
    need = int(frac * area)
    return max(cap[0], min(cap[1], need))

def roi_median_and_coverage(M, mask, x1, y1, x2, y2, min_valid=80):
    """
    Повертає (медіана по валідних пікселях у ROI, покриття_mask_в_ROI у
    [0..1]).
    Якщо mask=None, покриття = частка валідних пікселів самої матриці M.
    """
    if M is None:
        return float('nan'), 0.0
    H, W = M.shape[:2]
    x1, y1 = max(0, int(x1)), max(0, int(y1))
    x2, y2 = min(W - 1, int(x2)), min(H - 1, int(y2))
    if x2 <= x1 or y2 <= y1:
        return float('nan'), 0.0

    roi_M = M[y1:y2, x1:x2]
    if mask is not None:
        roi_mask = mask[y1:y2, x1:x2]
        valid_vals = roi_M[np.isfinite(roi_M) & roi_mask]
        coverage = float(np.count_nonzero(roi_mask)) / float(max(1,
roi_mask.size))
    else:
        valid_vals = roi_M[np.isfinite(roi_M)]
        coverage = float(valid_vals.size) / float(max(1, roi_M.size))

    if valid_vals.size >= min_valid:
        return float(np.median(valid_vals)), coverage
    return float('nan'), coverage

def roi_median_disparity(disparity, x1, y1, x2, y2, min_valid=80):
    """Медіана disparity у ROI (px)."""
    H, W = disparity.shape[:2]
    x1, y1 = max(0, int(x1)), max(0, int(y1))
    x2, y2 = min(W - 1, int(x2)), min(H - 1, int(y2))
    if x2 <= x1 or y2 <= y1:
        return float('nan')
    roi = disparity[y1:y2, x1:x2].astype(np.float32)
    roi = roi[np.isfinite(roi) & (roi > 0)]
    if roi.size >= min_valid:
        return float(np.median(roi))
    return float('nan')

def draw_box_with_metrics(img, x1, y1, x2, y2, z_pred, z_gt, absrel):
    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 220, 0), 2)
    if np.isfinite(z_pred) and np.isfinite(z_gt) and np.isfinite(absrel):
        label = f"pred={z_pred:.2f}m | gt={z_gt:.2f}m | rel={absrel:.3f}"

```

```

elif np.isfinite(z_pred):
    label = f"pred={z_pred:.2f}m | gt=n/a"
else:
    label = "n/a"
cv2.putText(img, label, (x1, max(0, y1 - 8)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 220, 0), 2)

def safe_detect(detector, frame_bgr, score_thr=SCORE_THR):
    """Підтримка обох версій ObjectDetector.detect."""
    try:
        return detector.detect(frame_bgr, score_thr=score_thr)
    except TypeError:
        dets = detector.detect(frame_bgr)
        if dets is None:
            return []
        out = []
        for d in dets:
            s = d.get("score", 1.0)
            if float(s) >= score_thr:
                out.append(d)
        return out

def list_kitti_indices(kitti_root):
    """Збирає всі базові імена файлів з image_2/*.png."""
    paths = sorted(glob.glob(os.path.join(kitti_root, "image_2", "*.png")))
    # повертаємо без .png і шляху: '000000_10', ...
    return [os.path.splitext(os.path.basename(p))[0] for p in paths]

# ===== Обробка одного індексу =====
def process_index(detector, idx):
    """
    Готує все для одного кадру:
    - читає L,R, fx, B, GT;
    - рахує disparity/depth (pred), depth_gt, gt_mask;
    - запускає детектор;
    - повертає всі візуалізації й таблицю результатів для CSV.
    """
    # 1) Дані KITTI + калібрування
    L, R, fx, B, gt_path = load_kitti_pair(KITTI_ROOT, idx)
    print(f"\n=== IDX {idx} | fx={fx:.1f} px, B={B:.3f} m, gt={'yes' if
gt_path else 'no'} ===")

    # 2) Stereo (KITTI вже ректифікований)
    stereo = StereoDepthEstimator(StereoCalib(fx=fx, baseline_m=B),
use_uncalibrated_rectify=False)

    # 3) Disparity → depth (pred)
    t0 = time.time()
    disp_pred = stereo.disparity(L, R)
    t_stereo = time.time() - t0

    d = disp_pred.astype(np.float32)
    d[d < 1.0] = np.nan
    depth_pred = depth_from_disp(d, fx, B)

```

```

# 4) GT disparity → depth_gt + маска покриття
depth_gt = None
gt_mask = None
if gt_path:
    d_gt = load_gt_disparity_png16(gt_path) # бажано disp_occ_0/
    if d_gt is not None:
        depth_gt = depth_from_disp(d_gt, fx, B)
        gt_mask = valid_mask_from_disparity(d_gt)
    else:
        print("[warn] GT disparity not loaded")

# 5) Детектор
t0 = time.time()
dets = safe_detect(detector, L, score_thr=SCORE_THR)
t_det = time.time() - t0
print(f"[time] stereo={t_stereo*1000:.1f}ms,
detector={t_det*1000:.1f}ms")
print(f"[det] {len(dets)} boxes with score >= {SCORE_THR}")

# 6) Візуалізація disparity
disp_vis = np.nan_to_num(d)
disp_vis = cv2.normalize(disp_vis, None, 0, 255,
cv2.NORM_MINMAX).astype("uint8")
disp_color = cv2.applyColorMap(disp_vis, cv2.COLORMAP_TURBO)

out = L.copy()
H, W = L.shape[:2]

# 7) Оцінка по всьому кадру (якщо є GT)
frame_metrics = {}
if depth_gt is not None and gt_mask is not None:
    frame_metrics = compute_depth_metrics(depth_gt, depth_pred,
mask=gt_mask)
    m = frame_metrics
    print(f"[metrics] AbsRel={m['abs_rel']:.3f} | RMSE={m['rmse']:.3f} |
d1={m['delta1']:.3f} | SILog={m['silog']:.2f}")

# 8) Оцінка по боксах
results_rows = [] # для CSV поточного кадру
valid_absrel = [] # для summary по кадру
any_printed = False

for i, det in enumerate(dets):
    ymin, xmin, ymax, xmax = det["box"]
    x1, y1 = int(xmin * W), int(ymin * H)
    x2, y2 = int(xmax * W), int(ymax * H)
    mvalid = adaptive_min_valid(x1, y1, x2, y2)

    d_med = roi_median_disparity(d, x1, y1, x2, y2, min_valid=mvalid)
    z_pred, _ = roi_median_and_coverage(depth_pred, None, x1, y1, x2,
y2, min_valid=mvalid)
    z_gt, cov_gt = roi_median_and_coverage(depth_gt, gt_mask, x1, y1, x2,
y2, min_valid=mvalid) if depth_gt is not None else (float('nan'), 0.0)

    absrel = float('nan')

```



```

        if np.isfinite(z_pred) and np.isfinite(z_gt) and cov_gt >=
GT_COVERAGE_THR and np.isfinite(d_med) and d_med >= MIN_MEDIAN_DISP:
            absrel = abs(z_pred - z_gt) / (z_gt + 1e-6)
            valid_absrel.append(absrel)

        print(f"[eval] box#{i} id={det.get('class_id','?')}
score={det.get('score',0):.2f} "
            f"-> pred={z_pred:.2f} m | gt={z_gt if np.isfinite(z_gt) else
float('nan'):.2f} m "
            f"| covGT={cov_gt*100:.1f}% | d_med={d_med if
np.isfinite(d_med) else float('nan'):.2f} px "
            f"| AbsRel={absrel if np.isfinite(absrel) else
float('nan'):.3f}")

        draw_box_with_metrics(out, x1, y1, x2, y2, z_pred, z_gt, absrel)
        any_printed = True

    row = {
        "idx": idx,
        "box_id": i,
        "class_id": det.get("class_id", "?"),
        "score": f"{det.get('score',0):.3f}",
        "x1": x1, "y1": y1, "x2": x2, "y2": y2,
        "covGT": f"{cov_gt:.3f}",
        "d_med": f"{d_med if np.isfinite(d_med) else float('nan'):.3f}",
        "z_pred": f"{z_pred if np.isfinite(z_pred) else
float('nan'):.3f}",
        "z_gt": f"{z_gt if np.isfinite(z_gt) else float('nan'):.3f}",
        "absrel": f"{absrel if np.isfinite(absrel) else
float('nan'):.3f}"
    }
    # Додаємо метрики всього кадру в кожен рядок для зручності
    if frame_metrics:
        for k, v in frame_metrics.items():
            row[f"frame_{k}"] = f"{v:.4f}"
    results_rows.append(row)

# 9) Зведення по кадру
if len(valid_absrel) > 0:
    arr = np.array(valid_absrel, dtype=np.float32)
    print(f"[summary:{idx}] used={len(arr)} | AbsRel
mean={arr.mean():.3f} | median={np.median(arr):.3f}")
else:
    print(f"[summary:{idx}] no boxes passed coverage & disparity
thresholds")

    return L, R, disp_color, out, results_rows, depth_pred, depth_gt, (d, ),
frame_metrics, (t_stereo, t_det)

def write_csv(rows, path):
    if not rows:
        return
    fieldnames = list(rows[0].keys())
    with open(path, "w", newline="") as f:
        w = csv.DictWriter(f, fieldnames=fieldnames)

```

```

        w.writeheader()
        for r in rows:
            w.writerow(r)
    print(f"[write] saved {len(rows)} rows -> {path}")

def main():
    # Збираємо список індексів
    indices = list_kitti_indices(KITTI_ROOT)
    if not indices:
        raise RuntimeError("Не знайдено файлів у image_2/*.png")

    # Модель детектора завантажуюємо один раз
    detector = ObjectDetector("models/ssd_mobilenet_v2")

    # Стан перегляду
    pos = 0
    all_rows = [] # акумулятор на 'W'
    click_img = None

    # Статистика по всьому сеансу
    session_metrics = []
    session_times = []

    # Початковий кадр
    L, R, disp_color, out, rows, depth_pred, depth_gt, extras, f_metrics,
    f_times = process_index(detector, indices[pos])
    all_rows.extend(rows)
    if f_metrics: session_metrics.append(f_metrics)
    session_times.append(f_times)
    click_img = out.copy()

    # Обробник кліку мишею для поточного кадру
    def on_mouse(event, x, y, flags, param):
        nonlocal click_img
        if event == cv2.EVENT_LBUTTONDOWN:
            x1, y1 = x - CLICK_HALF, y - CLICK_HALF
            x2, y2 = x + CLICK_HALF, y + CLICK_HALF
            # беремо disparity з extras[0] - це сирий disp (з NaN для <1px)
            d = extras[0]
            mvalid = adaptive_min_valid(x1, y1, x2, y2, cap=(20, 200))
            z_pred, _ = roi_median_and_coverage(depth_pred, None, x1, y1,
            x2, y2, min_valid=mvalid)
            z_gt, cov_gt = roi_median_and_coverage(depth_gt,
            valid_mask_from_disparity(depth_gt if depth_gt is None else depth_pred) if
            False else
            valid_mask_from_disparity(load_gt_disparity_png16(os.path.join(KITTI_ROOT,
            "disp_occ_0", f"{indices[pos]}.png")) if
            os.path.exists(os.path.join(KITTI_ROOT, "disp_occ_0", f"{indices[pos]}.png"))
            else None, x1, y1, x2, y2, min_valid=mvalid) if depth_gt is not None else
            (float('nan'), 0.0)
            d_med = roi_median_disparity(d, x1, y1, x2, y2, min_valid=mvalid)
            absrel = (abs(z_pred - z_gt) / (z_gt + 1e-6)) if
            np.isfinite(z_pred) and np.isfinite(z_gt) and cov_gt >= GT_COVERAGE_THR and
            np.isfinite(d_med) and d_med >= MIN_MEDIAN_DISP else float('nan')
            click_img = out.copy()

```

```

        draw_box_with_metrics(click_img, x1, y1, x2, y2, z_pred, z_gt,
absrel)

        print(f"[click:{indices[pos]}] ({x},{y}) -> pred={z_pred:.2f} m |
gt={z_gt if np.isfinite(z_gt) else float('nan'):.2f} m "
            f"| covGT={cov_gt*100:.1f}% | d_med={d_med if
np.isfinite(d_med) else float('nan'):.2f} px "
            f"| AbsRel={absrel if np.isfinite(absrel) else
float('nan'):.3f}")

cv2.namedWindow("Left (pred vs GT)")
cv2.setMouseCallback("Left (pred vs GT)", on_mouse)

while True:
    cv2.imshow("Left (pred vs GT)", click_img)
    cv2.imshow("Right", R)
    cv2.imshow("Disparity (pred)", disp_color)
    key = cv2.waitKey(10) & 0xFF

    if key in (ord('q'), ord('0')):    # вихід
        break

    elif key == ord('w'):                # запис CSV по поточному кадру
        write_csv(rows, CSV_PATH_PER_FRAME)

    elif key == ord('W'):                # запис CSV по всіх уже
переглянутих
        write_csv(all_rows, CSV_PATH_ALL)
        if session_metrics:
            print("\n=== SESSION SUMMARY METRICS ===")
            for k in session_metrics[0].keys():
                vals = [m[k] for m in session_metrics if
np.isfinite(m[k])]
                if vals:
                    print(f"{k:10s}: mean={np.mean(vals):.4f},
median={np.median(vals):.4f}")
            if session_times:
                t_stereo = [t[0] for t in session_times]
                t_det = [t[1] for t in session_times]
                print(f"Time Stereo: mean={np.mean(t_stereo)*1000:.1f}ms,
FPS={1.0/np.mean(t_stereo):.1f}")
                print(f"Time Detector: mean={np.mean(t_det)*1000:.1f}ms,
FPS={1.0/np.mean(t_det):.1f}")

    elif key == 32:
        pos = (pos + 1) % len(indices)
        # перерахунок для нового індексу
        L, R, disp_color, out, rows, depth_pred, depth_gt, extras,
f_metrics, f_times = process_index(detector, indices[pos])
        all_rows.extend(rows)
        if f_metrics: session_metrics.append(f_metrics)
        session_times.append(f_times)
        click_img = out.copy()
        # перевстановити callback (на нові об'єкти depth_pred/depth_gt)
        cv2.setMouseCallback("Left (pred vs GT)", on_mouse)
    elif key == ord('p'):

```

```

        pos = (pos - 1) % len(indices)
        # перерахунок для нового індексу
        L, R, disp_color, out, rows, depth_pred, depth_gt, extras,
f_metrics, f_times = process_index(detector, indices[pos])
        all_rows.extend(rows)
        if f_metrics: session_metrics.append(f_metrics)
        session_times.append(f_times)
        click_img = out.copy()
        # перевстановити callback (на нові об'єкти depth_pred/depth_gt)
        cv2.setMouseCallback("Left (pred vs GT)", on_mouse)

cv2.destroyAllWindows()

if __name__ == "__main__":
    main()

```

## Скрипт оцінки відстані за калібрувальними маркерами (ArUco) та задачею визначення пози (PnP)

```
# singlecam_aruco_distance.py
import cv2
import numpy as np
import time
import os

# ===== НАЛАШТУВАННЯ =====
CAM_INDEX = 0          # фронтальна/основна; на Mac інколи 0 або 1
FRAME_W, FRAME_H = 1280, 720
FPS = 30

# ТВОЯ ВИМІРЯНА СТОРОНА МІТКИ (МЕТРИ): 5.7 см = 0.057 м
MARKER_LENGTH_M = 0.057

INTR_FILE = "camera_intrinsics.npz" # якщо є K і dist
ASSUME_HFOV_DEG = 60.0             # якщо інтрінзиків нема — приблизна
оцінка

ARUCO_DICT = cv2.aruco.DICT_5X5_100
# Якщо друкував конкретний ID — вкажи його; якщо ні, то скрипт покаже будь-
який знайдений
PREFERRED_ID = None # наприклад 23, або залишити None

# =====

def approx_intrinsics(w, h, hfov_deg=60.0):
    fx = 0.5 * w / np.tan(np.deg2rad(hfov_deg) * 0.5)
    fy = fx
    cx, cy = w / 2.0, h / 2.0
    K = np.array([[fx, 0, cx],
                  [0, fy, cy],
                  [0, 0, 1]], dtype=np.float32)
    dist = np.zeros(5, np.float32)
    return K, dist

def load_intrinsics_or_guess(w, h):
    if os.path.exists(INTR_FILE):
        data = np.load(INTR_FILE)
        K = data["K"].astype(np.float32)
        dist = data["dist"].astype(np.float32)
        print(f"[calib] loaded intrinsics from {INTR_FILE}")
        return K, dist
    else:
        K, dist = approx_intrinsics(w, h, ASSUME_HFOV_DEG)
        print(f"[calib] using approximate intrinsics
(HFOV≈{ASSUME_HFOV_DEG}°): fx={K[0,0]:.1f}, cx={K[0,2]:.1f}")
        return K, dist

def pose_from_marker_corners(corners_xy, marker_length_m, K, dist):
    """
    corners_xy: np.ndarray shape (4,2) в ПОРЯДКУ ArUco:
```

```

    TL=(x0,y0), TR=(x1,y1), BR=(x2,y2), BL=(x3,y3)
    Рахуємо позу планарного квадрата через solvePnP.
    """
    s = float(marker_length_m)
    # 3D-кути в системі мітки (площина Z=0), центр – середина квадрата
    # ВАЖЛИВО: порядок має відповідати порядку із detectMarkers!
    objp = np.array([
        [-s/2, s/2, 0.0], # TL
        [ s/2, s/2, 0.0], # TR
        [ s/2, -s/2, 0.0], # BR
        [-s/2, -s/2, 0.0], # BL
    ], dtype=np.float32)

    imgp = corners_xy.astype(np.float32)

    # Найкраще для квадратних маркерів – IPPE_SQUARE (якщо доступно)
    flag = getattr(cv2, "SOLVEPNP_IPPE_SQUARE", None)
    if flag is None:
        flag = cv2.SOLVEPNP_ITERATIVE

    ok, rvec, tvec = cv2.solvePnP(objp, imgp, K, dist, flags=flag)
    if not ok:
        # fallback: спробуємо ITERATIVE якщо IPPE не зійшлась
        if flag != cv2.SOLVEPNP_ITERATIVE:
            ok2, rvec2, tvec2 = cv2.solvePnP(objp, imgp, K, dist,
            flags=cv2.SOLVEPNP_ITERATIVE)
            if ok2:
                return rvec2, tvec2
        return None, None
    return rvec, tvec

def main():
    cap = cv2.VideoCapture(CAM_INDEX)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FRAME_W)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_H)
    cap.set(cv2.CAP_PROP_FPS, FPS)
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

    if not cap.isOpened():
        raise RuntimeError("Cannot open camera")

    # Прочитаємо один кадр, щоб знати точний розмір
    ok, frame = cap.read()
    if not ok:
        raise RuntimeError("No frames from camera")
    h, w = frame.shape[:2]
    K, dist = load_intrinsics_or_guess(w, h)

    dictionary = cv2.aruco.getPredefinedDictionary(ARUCO_DICT)

    # OpenCV 4.7+ API:
    try:
        params = cv2.aruco.DetectorParameters()
        detector = cv2.aruco.ArucoDetector(dictionary, params)
        new_api = True

```

```

except Exception:
    # Стрипши API:
    params = cv2.aruco.DetectorParameters_create()
    detector = None
    new_api = False
    print("[info] Falling back to old detectMarkers API")

print("[info] Controls: q/0 = quit")
print(f"[info] Marker length set to {MARKER_LENGTH_M*100:.1f} cm")

fps_ema = None
while True:
    t0 = time.time()
    ok, frame = cap.read()
    if not ok:
        time.sleep(0.005)
        continue

    # Детекція кутів міток
    if new_api:
        corners_list, ids, _ = detector.detectMarkers(frame)
    else:
        corners_list, ids, _ = cv2.aruco.detectMarkers(frame, dictionary,
parameters=params)

    out = frame.copy()

    if ids is not None and len(ids) > 0:
        # Якщо задано PREFERRED_ID — фільтруємо
        if PREFERRED_ID is not None:
            filt = [i for i, mid in enumerate(ids.flatten()) if int(mid)
== int(PREFERRED_ID)]
        else:
            filt = list(range(len(ids)))

        for idx in filt:
            corners = corners_list[idx].reshape(-1, 2) # (4,2):
TL, TR, BR, BL

            rvec, tvec = pose_from_marker_corners(corners,
MARKER_LENGTH_M, K, dist)
            # Оформлення
            cv2.aruco.drawDetectedMarkers(out, [corners_list[idx]],
ids[idx])

            if rvec is not None and tvec is not None:
                # Відстань: Z-компонента (вперед) та повна норма
                t = tvec.reshape(3)
                z_forward = float(t[2])
                dist_3d = float(np.linalg.norm(t))
                # Вісь координат на мітці
                cv2.drawFrameAxes(out, K, dist, rvec, tvec,
MARKER_LENGTH_M * 0.5)

                x1, y1 = corners[:,0].min(), corners[:,1].min()
                label = f"id={int(ids[idx])} Z={z_forward:.2f} m |
|t|={dist_3d:.2f} m"

```

```

        cv2.putText(out, label, (int(x1), int(max(0, y1 - 8))),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255),
2)

        print(f"[dist] id={int(ids[idx])} -> Z={z_forward:.3f} m
| |t|={dist_3d:.3f} m")
    else:
        print("[warn] solvePnP failed on detected marker")

    # FPS/легенда
    dt = time.time() - t0
    fps_ema = (0.9 * fps_ema + 0.1 * (1.0 / dt)) if fps_ema else (1.0 /
dt)

    cv2.putText(out, f"fps={fps_ema:.1f}", (10, 26),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (50,220,255), 2)
    cv2.putText(out, "Controls: q/0 = quit", (10, 52),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200,200,200), 2)

    cv2.imshow("Single-cam ArUco distance (PnP)", out)
    key = cv2.waitKey(1) & 0xFF
    if key in (ord('q'), ord('0')):
        break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()

```



## Скрипт з використанням моноглибини TensorFlow та переходу до оцінки відстані

```
#main_mono_eval.py
import cv2
import numpy as np
import time

from models.depth_estimator_mono import MonoDepthEstimator
from models.detector import ObjectDetector
from datasets.kitti_loader import load_kitti_pair
from utils.metrics import compute_depth_metrics
from main_stereo import load_gt_disparity_png16, valid_mask_from_disparity,
depth_from_disp, safe_detect, adaptive_min_valid, roi_median_disparity,
roi_median_and_coverage, draw_box_with_metrics, list_kitti_indices, write_csv

# === Налаштування ===
KITTI_ROOT = "datasets/kitti2015/training" # шлях до training/
SCORE_THR = 0.5
GT_COVERAGE_THR = 0.20 # мін. частка валідних GT пікселів у ROI (0..1)
# для метрики
MIN_MEDIAN_DISP = 2.0 # мін. медіанна disparity в ROI (px), щоб не брати
# «наддалекі»
CLICK_HALF = 30 # півсторона ROI при кліку (60x60)
CSV_PATH_ALL = "mono_eval_all.csv"

# ===== Обробка одного індексу =====
def process_index(detector, estimator, idx, verbose=True):
    # 1) Дані KITTI
    L, R, fx, B, gt_path = load_kitti_pair(KITTI_ROOT, idx)
    if verbose:
        print(f"\n=== IDX {idx} | fx={fx:.1f} px, B={B:.3f} m, gt={'yes' if
gt_path else 'no'} ===")

    # 2) Mono Depth Inference (disparity-like)
    t0 = time.time()
    disp_pred = estimator.estimate(L)
    t_mono = time.time() - t0

    # 3) GT disparity -> depth_gt
    depth_gt = None
    gt_mask = None
    if gt_path:
        d_gt = load_gt_disparity_png16(gt_path)
        if d_gt is not None:
            depth_gt = depth_from_disp(d_gt, fx, B)
            gt_mask = valid_mask_from_disparity(d_gt)

    # 4) Вирівнювання масштабу (Scale alignment)
    # Оскільки моно-глибина відносна, для метрик її треба підігнати під GT.
    # Ми використовуємо медіанне вирівнювання: scale = median(gt_disp) /
median(pred_disp)
    if depth_gt is not None and gt_mask is not None:
        valid_pred_disp = disp_pred[gt_mask]
```

```

valid_gt_disp = (fx * B) / (depth_gt[gt_mask] + 1e-6)

# Вирівнюємо диспаратність
scale = np.median(valid_gt_disp) / (np.median(valid_pred_disp) + 1e-
6)

disp_aligned = disp_pred * scale
depth_pred = (fx * B) / (disp_aligned + 1e-6)
else:
    # Без GT просто фіксований масштаб для візуалізації
    depth_pred = 100.0 / (disp_pred + 1e-6)
    disp_aligned = disp_pred # для d_med

# 5) Детектор
t0 = time.time()
dets = safe_detect(detector, L, score_thr=SCORE_THR)
t_det = time.time() - t0
if verbose:
    print(f"[time] mono_depth={t_mono*1000:.1f}ms,
detector={t_det*1000:.1f}ms")

# 6) Візуалізація disparity
disp_vis = cv2.normalize(disp_pred, None, 0, 255,
cv2.NORM_MINMAX).astype("uint8")
disp_color = cv2.applyColorMap(disp_vis, cv2.COLORMAP_TURBO)

out = L.copy()
H, W = L.shape[:2]

# 7) Оцінка по всьому кадру
frame_metrics = {}
if depth_gt is not None and gt_mask is not None:
    frame_metrics = compute_depth_metrics(depth_gt, depth_pred,
mask=gt_mask)
    m = frame_metrics
    if verbose:
        print(f"[metrics] AbsRel={m['abs_rel']:.3f} |
RMSE={m['rmse']:.3f} | dl={m['delta1']:.3f}")

# 8) Оцінка по боксах
results_rows = []
valid_absrel = []

for i, det in enumerate(dets):
    ymin, xmin, ymax, xmax = det["box"]
    x1, y1 = int(xmin * W), int(ymin * H)
    x2, y2 = int(xmax * W), int(ymax * H)
    mvalid = adaptive_min_valid(x1, y1, x2, y2)

    # Використовуємо вирівняну диспаратність для d_med
    d_med = roi_median_disparity(disp_aligned if 'disp_aligned' in
locals() else disp_pred, x1, y1, x2, y2, min_valid=mvalid)
    z_pred, _ = roi_median_and_coverage(depth_pred, None, x1, y1, x2,
y2, min_valid=mvalid)
    z_gt, cov_gt = roi_median_and_coverage(depth_gt, gt_mask, x1, y1, x2,
y2, min_valid=mvalid) if depth_gt is not None else (float('nan'), 0.0)

```

```

        absrel = float('nan')
        if np.isfinite(z_pred) and np.isfinite(z_gt) and cov_gt >=
GT_COVERAGE_THR:
            absrel = abs(z_pred - z_gt) / (z_gt + 1e-6)
            valid_absrel.append(absrel)

        if verbose:
            print(f"[eval] box#{i} id={det.get('class_id','?')}
score={det.get('score',0):.2f} "
                f"-> pred={z_pred:.2f} m | gt={z_gt:.2f} m |
covGT={cov_gt*100:.1f}% "
                f"| d_med={d_med:.2f} px | AbsRel={absrel:.3f}")

        draw_box_with_metrics(out, x1, y1, x2, y2, z_pred, z_gt, absrel)

        row = {
            "idx": idx, "box_id": i, "class_id": det.get("class_id", "?"),
            "score": f"{det.get('score',0):.3f}",
            "z_pred": f"{z_pred:.3f}", "z_gt": f"{z_gt:.3f}", "absrel":
f"{absrel:.3f}",
            "covGT": f"{cov_gt:.3f}", "d_med": f"{d_med:.3f}"
        }
        if frame_metrics:
            for k, v in frame_metrics.items(): row[f"frame_{k}"] = f"{v:.4f}"
        results_rows.append(row)

    return L, R, disp_color, out, results_rows, depth_pred, depth_gt,
(disp_pred, ), frame_metrics, (t_mono, t_det)

def main():
    indices = list_kitti_indices(KITTI_ROOT)
    detector = ObjectDetector("models/ssd_mobilenet_v2")
    estimator = MonoDepthEstimator(model_type="MiDaS_small")

    pos = 0
    all_rows = []
    session_metrics = []
    session_times = []

    print("[controls] SPACE/->: next | <-: prev | W: save and show summary | q:
quit")

    last_idx = None

    while True:
        idx = indices[pos]
        verbose = (idx != last_idx)
        L, R, disp_color, out, rows, depth_pred, depth_gt, extras, f_metrics,
f_times = process_index(detector, estimator, idx, verbose=verbose)

        last_idx = idx

        cv2.imshow("Mono Depth Eval (aligned scale)", out)
        cv2.imshow("Raw Disparity", disp_color)

```

```

# Для автоматизації тестування ми можемо використати waitKey з
невеликим часом
# але тут залишимо 0 для ручного перегляду, або імітуємо натискання
key = cv2.waitKey(0) & 0xFF
if key in (ord('q'), ord('0')): break
elif key == 32:
    all_rows.extend(rows)
    if f_metrics: session_metrics.append(f_metrics)
    session_times.append(f_times)
    pos = (pos + 1) % len(indices)
elif key == ord('p'):
    all_rows.extend(rows)
    if f_metrics: session_metrics.append(f_metrics)
    session_times.append(f_times)
    pos = (pos - 1) % len(indices)
elif key == ord('W'):
    write_csv(all_rows, CSV_PATH_ALL)
    if session_metrics:
        print("\n=== SESSION SUMMARY ===")
        for k in session_metrics[0].keys():
            vals = [m[k] for m in session_metrics if
np.isfinite(m[k])]
            if vals: print(f"{k:10s}: mean={np.mean(vals):.4f}")
    if session_times:
        tm = [t[0] for t in session_times]; td = [t[1] for t in
session_times]
        print(f"Time Mono: {np.mean(tm)*1000:.1f}ms, Det:
{np.mean(td)*1000:.1f}ms")

    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()

```