

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного
аналізу Кафедра штучного інтелекту**

До захисту допущено:

В. о. завідувачки кафедри

_____Ірина ДЖИГИРЕЙ

«___»_____20__р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи
штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Методи побудови карт глибини та уникнення
перешкод для наземних дронів»**

Виконав:

студент IV курсу, групи КІ-22

Козинський Нікіта Олексійович

Керівник:

асистент кафедри штучного інтелекту, к.т.н.,

Осауленко Вячеслав Миколайович

Консультант з нормоконтролю:

фахівець першої категорії кафедри штучного

інтелекту, к.т.н., доцент, Комариста Богдана

Миколаївна

Рецензент:

старший викладач кафедри математичних

методів системного аналізу, к.т.н.,

Савастьянов Володимир Володимирович

Засвідчую, що у цій дипломній
роботі немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

Київ – 2026 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____Ірина ДЖИГИРЕЙ

«15» січня 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Козинському Нікіті Олексійовичу

1. Тема роботи «Методи побудови карт глибини та уникнення перешкод для наземних дронів», керівник Осауленко Вячеслав Миколайович, асистент кафедри штучного інтелекту, затверджені наказом по НН ІПСА від «27» травня 2026 р. № 1944-с.
2. Термін подання студентом роботи «05» червня 2026 року.
3. Вихідні дані до роботи: набір стереозображень, отриманих з віртуальної стереопари, змодельованої в середовищі Webots; параметри калібрування стереопари (внутрішні та зовнішні); еталонні карти глибини відповідних сцен; модель наземного дрона та сцена з перешкодами для тестування алгоритму уникнення; обмеження на роботу системи в режимі реального часу на центральному процесорі.
4. Зміст роботи: теоретичні відомості, проєктування системи, програмна реалізація, експериментальне оцінювання, аналіз результатів.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.
6. Дата видачі завдання «02» лютого 2026 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за темою	20.04.2026	Виконано
2	Підготовка першого розділу	27.04.2026	Виконано
3	Підготовка другого розділу	04.05.2026	Виконано
4	Розробка програмного продукту	11.05.2026	Виконано
5	Підготовка третього розділу	18.05.2026	Виконано
6	Підготовка четвертого розділу	25.05.2026	Виконано
7	Підготовка презентації доповіді	01.06.2026	Виконано
8	Оформлення дипломної роботи	05.06.2026	Виконано

Студент

Нікіта КОЗИНСЬКИЙ

Керівник

В'ячеслав ОСАУЛЕНКО

РЕФЕРАТ

Дипломна робота: 73 с., 7 рис., 1 табл., 26 посилань, 1 додаток.

КАРТА ГЛИБИНИ, СТЕРЕОЗІР, ДИСПАРАТНІСТЬ, УНИКНЕННЯ ПЕРЕШКОД, АВТОНОМНА НАВІГАЦІЯ, НАЗЕМНИЙ ДРОН, SGBM, RAFT-STEREO, WEBOTS, MIDDLEBURY.

Метою роботи є розробка, програмна реалізація та експериментальне оцінювання системи стереозорової побудови карти глибини й реактивного уникнення перешкод для наземного дрона.

Об'єкт дослідження – процес автономної навігації наземного безпілотного апарата у середовищі з перешкодами.

Предмет дослідження – методи побудови карт глибини за даними бортової стереопари та алгоритми виявлення й уникнення перешкод.

У роботі проаналізовано сучасні підходи до побудови карт глибини та уникнення перешкод. Розроблено систему, що поєднує методи SGBM і RAFT-Stereo для оцінки глибини та реактивний алгоритм керування на основі скінченного автомата. Для дослідження роботи системи створено модель робота в симуляторі Webots, а оцінювання точності виконано на наборі даних Middlebury 2021.

Встановлено, що метод RAFT-Stereo забезпечує вищу точність відновлення глибини, тоді як SGBM потребує менших обчислювальних ресурсів і придатний до роботи в режимі реального часу. Проведені експерименти підтвердили працездатність запропонованого алгоритму уникнення перешкод.

ABSTRACT

Bachelor's thesis: 73 pages, 7 figures, 1 table, 26 references, 1 appendix.

DEPTH MAP, STEREO VISION, DISPARITY, OBSTACLE AVOIDANCE, AUTONOMOUS NAVIGATION, GROUND DRONE, SGBM, RAFT-STEREO, WEBOTS, MIDDLEBURY.

The purpose of this work is to develop, implement, and experimentally evaluate a stereo vision-based depth estimation and reactive obstacle avoidance system for a ground drone.

The object of research is the process of autonomous navigation of an unmanned ground vehicle in an environment with obstacles.

The subject of research includes depth estimation methods based on stereo camera data and obstacle avoidance algorithms.

The thesis analyzes modern approaches to depth estimation and obstacle avoidance. A system combining the SGBM and RAFT-Stereo methods with a finite-state-machine-based reactive controller has been developed. A robot model was created in the Webots simulator, and quantitative evaluation was performed using the Middlebury 2021 dataset.

The results show that RAFT-Stereo provides higher depth estimation accuracy, whereas SGBM requires fewer computational resources and is suitable for real-time operation. Experimental evaluation confirmed the effectiveness of the proposed obstacle avoidance algorithm.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ.....	8
ВСТУП	9
РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ АВТОНОМНОЇ НАВІГАЦІЇ НАЗЕМНИХ ДРОНІВ	12
1.1 Наземні дрони та задача автономної навігації	12
1.2 Карта глибини як представлення навколишнього простору	14
1.3 Сенсори для отримання інформації про глибину сцени	18
1.4 Класифікація методів побудови карт глибини.....	21
1.5 Підходи до уникнення перешкод та планування шляху	23
Висновки до розділу 1.....	25
РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ТА ОБҐРУНТУВАННЯ СТРУКТУРИ МОДЕЛІ	27
2.1 Геометричні основи побудови карт глибини	27
2.2 Класичні алгоритми стереозору та SLAM.....	30
2.3 Нейромережеві методи оцінки глибини	32
2.4 Обґрунтування вибору моделей для даної роботи.....	35
2.5 Метрики оцінювання якості карт глибини	38
Висновки до розділу 2.....	39
РОЗДІЛ 3 РОЗРОБКА МОДЕЛЕЙ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ	40
3.1 Загальна архітектура системи	40
3.2 Моделювання сцени та стереопари в середовищі Webots	42
3.3 Калібрування стереопари та геометричні перетворення	43
3.4 Побудова карти глибини зі стереопари.....	44
3.4.1 Класичний метод SGBM	45
3.4.2 Нейромережевий метод RAFT-Stereo	47
3.5 Виявлення перешкод за картою глибини.....	48
3.6 Алгоритм уникнення перешкод	50
Висновки до розділу 3.....	53

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.....	55
4.1 Методика проведення експериментів	55
4.2 Формування еталонних карт глибини	57
4.3 Порівняння якості карт глибини методами SGBM та RAFT-Stereo .	58
4.4 Дослідження роботи алгоритму уникнення перешкод.....	62
4.5 Аналіз результатів та обмеження системи	65
Висновки до розділу 4.....	67
ВИСНОВКИ	69
ПЕРЕЛІК ПОСИЛАНЬ.....	71
ДОДАТОК А ЛІСТИНГ КОДУ	74

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

UGV – Unmanned Ground Vehicle

SLAM – Simultaneous Localization and Mapping

FOV – Field of View

IMU – Inertial Measurement Unit

GPS – Global Positioning System

SOTA - State Of The Art

LiDAR – Light Detection and Ranging

RGB-D – Red, Green, Blue + Depth

ToF – Time-of-Flight

SGBM – Semi-Global Block Matching

RAFT-Stereo – Recurrent All-Pairs Field Transforms for Stereo

SAD – Sum of Absolute Differences

DPT – Dense Prediction Transformer

AbsRel – Absolute Relative error

RMSE – Root Mean Square Error

SPL – Success weighted by Path Length

ROS – Robot Operating System

Nav2 – Navigation 2

DWA – Dynamic Window Approach

APF – Artificial Potential Fields

RRT – Rapidly-exploring Random Tree

ВСТУП

Наземні¹ безпілотні платформи перестали бути лабораторною екзотикою. Сьогодні їх застосовують у складській логістиці, інспекції промислових об'єктів, агросекторі, розмінуванні та військовій розвідці. Український досвід останніх років показав, що навіть прості колісні платформи з елементами автономності здатні виконувати завдання, які раніше потребували присутності людини в зоні підвищеного ризику.

Будь-яка автономна дія наземного дрона починається з того, що машина має зрозуміти простір навколо себе. Без коректного 3D-представлення сцени неможливо безпечно рухатися: дрон або зупиниться перед уявною перешкодою, або, що гірше, не помітить реальну. Карта глибини є одним з ключових проміжних представлень, яке перетворює дані з камери чи іншого сенсора у форму, придатну для подальшого планування шляху.

Задача побудови карти глибини не є новою. Класичні методи стереозору відомі з 80-х років, а LiDAR-сенсори давно стали стандартом у автомобільній індустрії. Проте за останні кілька років ситуація суттєво змінилася. Поява моделей на основі архітектури Transformer (зокрема моделі монокулярної оцінки глибини MiDaS [3], Dense Prediction Transformer (DPT) [4], Depth Anything [5]) дозволила отримувати достатньо точні карти глибини навіть з однієї звичайної камери. Це принципово важливо для невеликих наземних платформ, де простір під сенсори та бюджет на обладнання обмежені.

¹ Тут і нижче використано такий(і) інструмент(и) штучного інтелекту як чат-бот з генеративним штучним інтелектом ChatGPT, виключно для корегування та редагування тексту, створеного автором цієї дипломної роботи, на основі автоматизованої перевірки граматики, структури та стилю, що відповідає Політиці використання штучного інтелекту для академічної діяльності в КПІ ім. Ігоря Сікорського (протокол №11 Вченої ради КПІ ім. Ігоря Сікорського від 11 грудня 2023 р.).

Паралельно розвивалися й алгоритми уникнення перешкод. Від простих реактивних методів на кшталт Bug-алгоритму індустрія перейшла до гібридних архітектур, у яких глобальний планувальник прокладає маршрут на карті, а локальний обробляє непередбачувані ситуації в реальному часі. Стандартом де-факто став програмний стек Robot Operating System 2 (ROS 2) [18] з пакетом навігації Navigation 2 (Nav2) [19], який поєднує обидва рівні.

Актуальність теми зумовлена кількома чинниками. По-перше, попит на автономні наземні платформи зростає швидше, ніж кількість фахівців, здатних їх розробляти й налаштовувати. По-друге, готові комерційні рішення часто є закритими або надто дорогими для дрібних задач, тому open-source стек на базі ROS 2 і відкритих моделей оцінки глибини є практичним напрямом. По-третє, методи глибокого навчання для оцінки глибини за останні два-три роки вийшли на рівень, коли їх уже доцільно інтегрувати в реальні системи навігації, а не лише досліджувати в лабораторних умовах.

Метою роботи є розробка, програмна реалізація та експериментальне оцінювання побудови карт глибини та алгоритмів уникнення перешкод для наземних дронів, а також формування теоретичної, математичної та технологічної основи для розробки відповідної системи в рамках дипломної роботи. Для досягнення цієї мети було проаналізовано класичні геометричні підходи, сучасні нейромережеві моделі оцінки глибини, алгоритми локального і глобального планування.

Об'єктом дослідження є процес автономної навігації наземного безпілотного апарата у середовищі з перешкодами.

Предметом дослідження є методи побудови карт глибини за даними бортових сенсорів та алгоритми планування шляху й уникнення перешкод.

Основними завданнями роботи є аналіз сфер застосування та особливостей наземних дронів як об'єкта автономної навігації; огляд методів побудови карт глибини; дослідження класичних і нейромережевих підходів до оцінки глибини; розгляд алгоритмів уникнення перешкод.

Практична цінність роботи полягає у можливості застосування результатів дослідження для побудови систем автономної навігації наземних безпілотних платформ, які орієнтовані на роботу в реальних умовах із обмеженим складом сенсорики та обчислювальних ресурсів.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ АВТОНОМНОЇ НАВІГАЦІЇ НАЗЕМНИХ ДРОНІВ

1.1 Наземні дрони та задача автономної навігації

Наземні безпілотні апарати (Unmanned Ground Vehicles) являють собою клас мобільних робототехнічних систем, призначених для виконання завдань без безпосередньої присутності людини на борту. На відміну від повітряних безпілотників, наземні платформи рухаються по поверхні і мають постійний контакт з опорою, що накладає свої вимоги до системи сприйняття та керування. Сьогодні UGV застосовують у складській логістиці, інспекції промислових об'єктів і ліній електропередач, обробці сільськогосподарських полів, патрулюванні охоронюваних територій, розмінуванні та задачах військового призначення.

За способом переміщення наземні дрони поділяють на колісні, гусеничні та крокуючі. Колісні платформи є найпоширенішими через простоту конструкції, високу енергоефективність і добру маневреність на твердих поверхнях. Гусеничні моделі забезпечують кращу прохідність на пересіченій місцевості, але споживають більше енергії і складніші в обслуговуванні. Крокуючі апарати (зокрема чотириногі роботи на кшталт Boston Dynamics Spot або Unitree Go2) демонструють найкращу адаптивність до складного рельєфу, проте поки що залишаються відносно дорогими і вимагають значно складніших алгоритмів керування.

За рівнем автономності виділяють кілька категорій наземних дронів. Найпростіші є телекерованими – оператор постійно контролює рух апарата через канал зв'язку. Напіваавтономні платформи можуть виконувати окремі дії самостійно (утримання курсу, повернення в точку старту, об'їзд найочевидніших перешкод), але загальну стратегію руху задає людина. Повністю автономні системи здатні самостійно планувати маршрут і досягати

поставленої цілі без зовнішнього втручання. Більшість сучасних комерційних і військових UGV мають базові елементи автономності, але рішення на верхньому рівні залишаються за оператором.

Перехід від телекерування до автономного режиму означає, що частину рішень про рух машина має ухвалювати самостійно. Це принципово змінює вимоги до бортової електроніки і програмного забезпечення. Якщо телекерований апарат у простому випадку може складатися з шасі, двигунів, камери та радіоканалу, то автономний потребує повноцінного навігаційного стеку: системи сприйняття, локалізації, планувальника шляху, контролера руху і модуля прийняття рішень. Усі ці компоненти мають працювати узгоджено і в реальному часі.

Навігаційний стек автономного дрона традиційно розв'язує три взаємопов'язані задачі: локалізацію, сприйняття навколишнього середовища та планування шляху. Локалізація полягає в оцінюванні власного положення апарата у просторі і реалізується за допомогою Global Positioning System (GPS), одометрії, інерціальних вимірювань або алгоритмів Simultaneous Localization and Mapping (SLAM). Сприйняття середовища спрямоване на побудову моделі навколишнього простору на основі даних бортових сенсорів, причому ключову роль у цьому процесі відіграє карта глибини або інше 3D-представлення сцени. Планування шляху використовує отриману модель середовища для побудови траєкторії до цілі з урахуванням перешкод і обмежень руху. Задача уникнення перешкод працює на стику другого і третього компонентів, оскільки потребує одночасно і коректного сприйняття простору, і здатності оперативно коригувати траєкторію в умовах динамічних змін.

Особливістю наземної навігації порівняно з повітряною є наявність обмежень з боку поверхні. Дрон не може просто «перелетіти» через перешкоду, тому навіть невеликий бордюр, яма чи купа сміття можуть зупинити рух. Це робить вимоги до точності сприйняття перешкод вищими,

ніж для квадрокоптера, який часто може обійти проблему по висоті. Окрім того, наземна сцена зазвичай складніша геометрично: вона містить більше близьких об'єктів, динамічних перешкод (люди, інші транспортні засоби) і неоднорідних поверхонь, які впливають на прохідність.

Окремою практичною обставиною є обмеженість бортових ресурсів. На відміну від автономного автомобіля, де можна встановити кілька кіловат обчислювальних потужностей і десятки кілограмів сенсорики, невеликий наземний дрон зазвичай має обмежений бюджет за вагою, споживанням енергії та обчислювальними можливостями. Це робить вибір методів сприйняття та планування не суто академічним, а інженерним питанням: необхідно знаходити баланс між точністю алгоритмів і їх обчислювальною складністю, аби вся система могла працювати в реальному часі на бортовому комп'ютері.

Таким чином, задача автономної навігації наземного дрона зводиться до побудови повного циклу, у якому центральною ланкою є коректне 3D-представлення навколишнього простору.

1.2 Карта глибини як представлення навколишнього простору

Карта глибини являє собою растрове представлення сцени, у якому кожному пікселю замість значення яскравості або кольору поставлено у відповідність значення відстані від камери до відповідної точки реального простору. Структурно вона ідентична звичайному двовимірному зображенню, проте її семантика принципово відрізняється: пікселі кодують геометричну, а не радіометричну інформацію.

Формально карту глибини можна задати як функцію

$$D: \Omega \rightarrow R^+, D(u, v) = Z,$$

де $\Omega \subset N^2$ – множина координат пікселів зображення, а Z – відстань від оптичного центра камери до точки сцени, видимої через піксель (u, v) . На практиці карта глибини зберігається у вигляді одноканального зображення з 16- або 32-бітною точністю на піксель, що дозволяє представляти значення відстаней від кількох сантиметрів до десятків метрів без втрати точності.

Карта глибини зручна для задач навігації з двох причин. Вона лишається двовимірним зображенням і допускає стандартні методи комп'ютерного зору, але водночас несе метричну інформацію про відстань до кожної точки сцени. Хмару точок або сітку зайнятості з неї можна отримати простим перетворенням за наявності калібрування камери, тоді як зворотні перетворення вже не такі прямолінійні.

Близьким за змістом, але не тотожним поняттям є карта диспаратності, яка використовується у стереозоровому підході до побудови глибини. Диспаратність визначається як горизонтальне зміщення відповідних точок між зображеннями з лівої та правої камер каліброваної стереопари. Зв'язок між диспаратністю та глибиною задається класичною формулою стереозору.

Карта глибини може бути перетворена у тривимірне представлення сцени за допомогою зворотної проєкції пікселів у простір камери. Для пінхольної моделі камери з матрицею внутрішніх параметрів, що містить фокусні відстані f_x, f_y і координати головної точки c_x, c_y , координати точки у просторі (X, Y, Z) , яка відповідає пікселю (u, v) зі значенням глибини $Z = D(u, v)$, обчислюються за формулами:

$$X = \frac{(u - c_x) \cdot Z}{f_x}$$

$$Y = \frac{(v - c_y) \cdot Z}{f_y}.$$

У такий спосіб карту глибини можна перетворити у хмару точок та, відповідно, у будь-яке інше просторове представлення сцени.

Розглянемо два характерні приклади карт глибини, отриманих у різних умовах знімання. На рисунку 1.1 наведено приклад внутрішньої сцени з набору даних NYU Depth V2 [17]: RGB-зображення типової житлової кімнати та відповідна карта глибини.



Рисунок 1.1 – Приклад внутрішньої сцени з набору NYU Depth V2

Внутрішні сцени характеризуються відносно невеликим діапазоном відстаней, наявністю великих плоских поверхонь, а також численними дрібними об'єктами на близькій відстані. Карти глибини в таких умовах зазвичай мають високу щільність. Водночас навіть для внутрішніх сцен карти глибини можуть містити області з відсутніми значеннями. Наприклад, через тіні від інфрачервоного випромінювача RGB-D-сенсора (Red, Green, Blue + Depth) або через відбивні поверхні з низьким альбедо.

Зовнішні сцени мають принципово інший характер: діапазон відстаней значно ширший, присутні великі області неба, для яких поняття глибини не визначене, а сама структура сцени містить більше дрібних деталей на середніх і дальніх дистанціях. Карти глибини для зовнішніх сцен зазвичай мають нижчу щільність валідних значень і вищу частку шуму, особливо на віддалених

об'єктах. На рисунку 1.2 показано приклад зовнішньої сцени з набору даних вуличного пейзажу, знятого з автомобільного дрона KITTI [16].

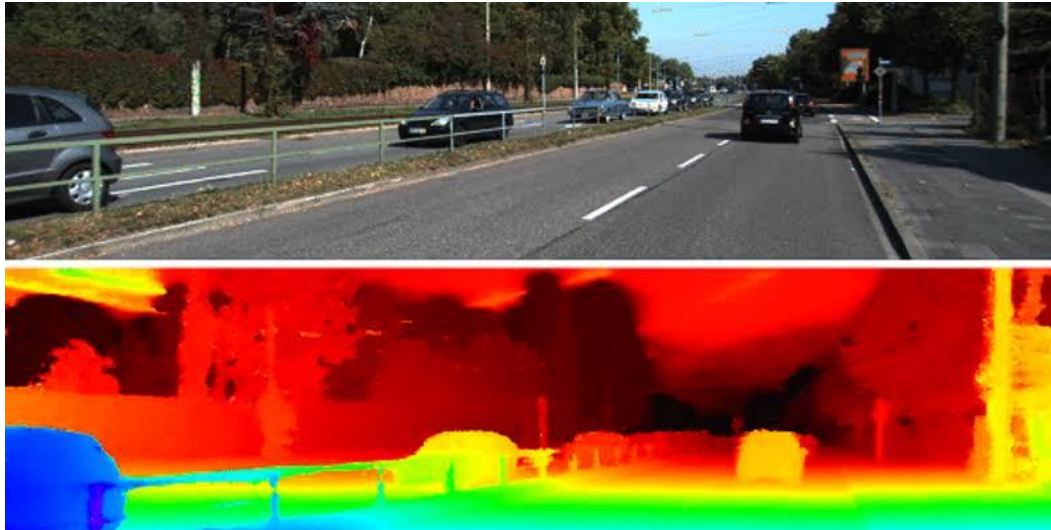


Рисунок 1.2 – Приклад зовнішньої сцени з набору KITTI

Порівняння двох прикладів демонструє важливий практичний висновок: якісні характеристики карти глибини суттєво залежать від умов знімання та типу сцени. Це впливає на вибір методів побудови глибини та на налаштування параметрів алгоритмів верхнього рівня. Для наземних дронів, які можуть працювати як в умовах закритих приміщень, так і у відкритому просторі, важливо забезпечити стійкість алгоритму оцінки глибини до обох типів сцен.

За способом подання значень розрізняють метричні та відносні карти глибини. Метричні карти містять реальні значення відстані у фізичних одиницях вимірювання. Їх можна безпосередньо використовувати для геометричних розрахунків, зокрема для оцінки висоти перешкод, дистанції до них та розмірів вільного простору. Відносні карти кодують лише порядок розташування об'єктів за глибиною. Іншими словами, вони показують, який об'єкт ближчий, а який далі, без точних значень відстані. Відносні карти часто

є результатом роботи нейромережових монокулярних моделей оцінки глибини, які не можуть однозначно визначити масштаб сцени за одним зображенням. Це явище відоме як *scale ambiguity*. Для практичного застосування в навігації відносні карти зазвичай потребують додаткового масштабування за допомогою даних з IMU, одометрії або відомих розмірів орієнтирів.

У навігаційному стеку наземного дрона карта глибини виконує роль проміжного представлення між сирими даними сенсора і модулями верхнього рівня. Камера або стереопара формує зображення, далі алгоритм оцінки глибини перетворює його на карту глибини, а вже потім ця карта використовується для побудови сітки зайнятості, виявлення перешкод, сегментації прохідних поверхонь або локалізації. Така проміжна позиція робить карту глибини центральним елементом системи сприйняття: якість усіх подальших етапів навігації безпосередньо залежить від того, наскільки точно і стабільно вона формується.

1.3 Сенсори для отримання інформації про глибину сцени

Якість карти глибини у системі автономної навігації значною мірою визначається тим, який сенсор використовується для збору первинних даних. Сьогодні існує кілька принципово різних класів сенсорів, кожен зі своїми перевагами та обмеженнями. Вибір конкретного типу залежить від умов експлуатації наземного дрона, доступного бюджету, обчислювальних ресурсів та вимог до точності й дальності вимірювань.

Монокулярна камера є найпростішим і найдешевшим джерелом візуальної інформації про сцену. Це звичайна цифрова камера з одним об'єктивом, яка фіксує двовимірне зображення. Сама собою монокулярна

камера не дає прямої інформації про відстань до об'єктів, оскільки під час проєкції тривимірної сцени на двовимірну площину інформація про глибину втрачається. Проте у поєднанні з алгоритмами комп'ютерного зору, зокрема з нейромережевими моделями оцінки глибини, монокулярна камера дозволяє формувати достатньо якісні карти глибини. До її переваг належать низька вартість, малі габарити, низьке енергоспоживання та широка доступність. Основним обмеженням є те, що глибина оцінюється алгоритмічно, а не вимірюється безпосередньо. Це означає, що монокулярні карти глибини часто є відносними і потребують додаткового масштабування для практичного застосування.

Стереокамера складається з двох рознесених у просторі камер, що знімають сцену одночасно. Принцип роботи базується на тригонометричному обчисленні відстані за різницею положень одних і тих самих точок сцени на двох зображеннях. Глибина в цьому випадку є геометричною величиною, що обчислюється безпосередньо за параметрами калібрування і не залежить від навчених моделей. Стереокамери забезпечують метричні карти глибини без проблеми *scale ambiguity*, характерної для монокулярних методів. Серед переваг варто відзначити прийнятну точність на близьких і середніх дистанціях, незалежність від зовнішнього освітлення для пасивних стереокамер та відсутність потреби в активному випромінюванні. Обмеженнями є чутливість до текстури сцени: на однорідних поверхнях алгоритм просто не може знайти відповідні точки. Точність помітно погіршується зі збільшенням дистанції, а сама пара камер потребує ретельного калібрування.

RGB-D сенсори поєднують звичайну RGB-камеру з активним джерелом інформації про глибину. Найчастіше для оцінки глибини застосовують структуроване підсвічування, коли камера проєктує на сцену відомий патерн інфрачервоного випромінювання і за його деформацією обчислює відстань. Альтернативою є принцип *Time-of-Flight*, про який детальніше йдеться нижче.

RGB-D сенсори видають синхронізовані пари кольорового зображення разом з картою глибини, що зручно для подальшої обробки. До переваг належать висока щільність карти глибини, метричні значення відстані та відносна простота інтеграції. Серед обмежень варто зазначити невелику робочу дальність для більшості моделей, залежність від умов освітлення і обмежену роботу на відбивних, прозорих та дуже темних поверхнях.

LiDAR є активним сенсором, який вимірює відстань до об'єктів за часом проходження лазерного імпульсу від випромінювача до об'єкта і назад. Сучасні LiDAR-сенсори бувають механічними з обертовим блоком випромінювача або твердотільними без рухомих частин. Простіші моделі сканують сцену лише в горизонтальній площині, тоді як багат шарові формують повноцінну тривимірну хмару точок. Вони забезпечують найвищу серед розглянутих сенсорів точність вимірювання відстаней та значну робочу дальність, яка для якісних моделей сягає сотень метрів. Він добре працює в умовах поганого освітлення і вночі, оскільки не залежить від зовнішнього світла. Основні обмеження пов'язані з високою вартістю, особливо для багат шарових моделей з великою кутовою роздільністю. Сенсор має значне енергоспоживання та чутливий до погодних умов: дощ, туман і сніг розсіюють лазерний промінь. Особливістю є те, що LiDAR не дає кольорової інформації, тому його часто комбінують з камерами для отримання повноцінного уявлення про сцену.

Окрім чотирьох основних типів сенсорів, у системах навігації наземних дронів іноді застосовують додаткові класи обладнання. ToF-камери працюють за тим самим принципом, що й LiDAR, але формують карту глибини відразу для всього кадру за допомогою модульованого світлового потоку. Вони дають щільні карти глибини на близьких дистанціях, проте мають обмежену дальність і чутливі до зовнішнього освітлення. Ультразвукові датчики використовують відлуння звукових хвиль для оцінки відстані до перешкоди. Вони дешеві, надійні та працюють у будь-яких умовах освітлення, але дають

лише точкові виміри без формування повноцінної карти глибини, а їхня дальність зазвичай обмежена кількома метрами. Радари використовують радіохвилі і добре працюють у складних погодних умовах, однак мають низьку кутову роздільність та переважно застосовуються для виявлення великих рухомих об'єктів, а не для побудови детальної карти оточення.

Спираючись на аналіз розглянутих типів сенсорів, стає зрозуміло, що жоден з них не є універсальним. Для невеликих наземних дронів, які повинні працювати в обмежених умовах за бюджетом і енергоспоживанням, найбільш привабливим є використання монокулярної або стереокамери у поєднанні з сучасними алгоритмами оцінки глибини. Для платформ із підвищеними вимогами до точності та дальності виявлення перешкод доцільним є застосування LiDAR. У реальних системах часто використовують комбінацію сенсорів різних типів, що дозволяє компенсувати слабкі сторони одного класу обладнання сильними сторонами іншого. Такий підхід відомий як сенсорний синтез, або *sensor fusion*, і детальніше розглядається в наступних розділах роботи.

1.4 Класифікація методів побудови карт глибини

Методи побудови карт глибини зазвичай класифікують за двома ознаками: типом вхідних даних і характером алгоритму. Перша ознака визначається обраним сенсором, друга – тим, як саме система переходить від сирих даних до значень глибини. Ці дві осі взаємно незалежні, тому будь-який конкретний метод можна описати парою з типу даних і типу алгоритму.

За типом вхідних даних виділяють чотири основні групи. Монокулярні методи працюють з одним зображенням від звичайної камери. Стереоскопічні методи використовують синхронну пару зображень з двох камер.

Багатоглядові методи розширюють стерео на три і більше ракурсів або на послідовні кадри рухомого сенсора. Окрему групу складають методи на основі активних сенсорів, зокрема LiDAR, RGB-D і ToF. У таких системах інформація про глибину частково надається самим обладнанням, а завданням алгоритму є її ущільнення або поєднання з даними від інших джерел.

За характером алгоритму методи поділяють на геометричні та нейромережеві. Геометричні методи обчислюють глибину через явні математичні залежності, такі як епіпольна геометрія, тріангуляція, рівняння руху камери. Вони добре інтерпретовні і не потребують навчальних даних, але чутливі до якості калібрування, наявності текстури в сцені та умов освітлення. Нейромережеві методи замість явних формул використовують моделі глибокого навчання, які відображають вхідні зображення у карту глибини на основі досвіду, набутого під час тренування. Вони краще справляються зі складними сценами, але потребують якісних навчальних наборів та значних обчислювальних ресурсів.

Поєднання двох осей дає кілька характерних родин методів. Монокулярні геометричні підходи зазвичай не працюють з одним кадром і потребують послідовності зображень, на основі якої будується модель руху камери і одночасно оцінюється глибина. Монокулярні нейромережеві методи навпаки здатні отримати карту глибини з єдиного зображення і за останні роки досягли якості, достатньої для практичних задач. Стереоскопічні геометричні методи зводяться до пошуку відповідних точок між двома зображеннями. Стереоскопічні нейромережеві методи вирішують ту саму задачу, але доручають пошук відповістей навченій моделі, що підвищує стійкість до однорідних поверхонь і поганого освітлення.

Окремо варто виділити гібридні підходи, у яких поєднуються дані від різних сенсорів або алгоритми різного типу. Найпоширенішим прикладом є об'єднання LiDAR і камери. LiDAR дає точні, але розріджені виміри, тоді як нейромережа на їх основі і на основі RGB-зображення формує щільну карту

глибини. Інший варіант полягає у поєднанні класичного візуального SLAM з нейромережевою оцінкою глибини для роботи в сценах із поганою текстурою. Такі рішення є більш складними з точки зору інтеграції, але дають кращі результати в реальних умовах експлуатації.

1.5 Підходи до уникнення перешкод та планування шляху

Після того як система сприйняття сформувала модель навколишнього простору, наступним кроком навігаційного стеку є перетворення цієї моделі у керуючі команди для руху. Цей процес зазвичай складається з двох рівнів. На верхньому рівні виконується глобальне планування маршруту від поточного положення дрона до цільової точки. На нижньому рівні працює модуль реактивного уникнення перешкод і локального коригування траєкторії, який обробляє ситуації, що не були враховані глобальним планом. Поділ задачі на ці два рівні є основою архітектури більшості сучасних навігаційних систем.

Глобальне планування передбачає, що дрон має доступ до певного представлення карти середовища, наприклад сітки зайнятості або графа прохідних областей. Найбільш поширеним класичним алгоритмом тут є A-star [11], який знаходить оптимальний шлях між двома точками за допомогою евристичного пошуку. Розширенням A-star для динамічних середовищ є алгоритм D-star, що дозволяє ефективно перераховувати шлях при появі нових перешкод. Окрему гілку методів складають алгоритми на основі випадкового вибірки, такі як RRT [12] і RRT-star [13]. Вони працюють в неперервному просторі станів і добре підходять для платформ з кінематичними обмеженнями. Перевагою RRT-методів є здатність швидко знаходити допустимий шлях у складних середовищах, а недоліком виступає те, що шлях не завжди є оптимальним без додаткового опрацювання.

Локальне планування і реактивне уникнення перешкод вирішують іншу задачу. Як рухатися в межах декількох секунд або метрів, маючи лише часткову інформацію про оточення. Один з найпростіших класичних підходів називається Bug-алгоритмами, які реалізують поведінку обходу перешкоди по її контуру до моменту, поки не з'явиться можливість продовжити рух у бік цілі. Більш практичним є метод штучних потенціальних полів (APF), у якому цільова точка створює притягальне поле, а перешкоди створюють відштовхувальне, і дрон рухається вздовж результуючого вектора. Поширеним недоліком APF є проблема локальних мінімумів, у яких дрон може застрягти між кількома перешкодами.

Серед локальних планувальників особливе місце посідає метод динамічного вікна Dynamic Window Approach [14]. Він безпосередньо враховує кінематичні та динамічні обмеження апарата і обирає пару керуючих команд лінійної та кутової швидкостей з простору допустимих значень. Кожен варіант руху оцінюється за функцією вартості, яка враховує близькість до глобального плану, дистанцію до перешкод і поточну швидкість. DWA добре зарекомендував себе для колісних платформ і є одним зі стандартних локальних планувальників у популярних навігаційних фреймворках. Близьким за духом методом є VFH [15], який будує гістограму густини перешкод навколо дрона і вибирає напрямок руху на основі цієї гістограми.

Окремо варто згадати методи на основі навчання з підкріпленням (Reinforcement Learning) та end-to-end нейромережових підходів до навігації. У цьому випадку правила прийняття рішень не задаються явно, а формуються в процесі навчання моделі на основі взаємодії з середовищем або з демонстрацій. Такі підходи демонструють хороші результати у складних і слабо структурованих сценах, де класичні алгоритми планування мають обмеження. Водночас вони потребують значних обчислювальних ресурсів для навчання, великих наборів навчальних даних або реалістичних симуляторів і

поки що менш надійні з погляду гарантованої безпеки руху, ніж класичні детерміновані методи.

У сучасних навігаційних системах рідко застосовують будь-який з перелічених методів окремо. Найпоширенішою архітектурою є гібридне поєднання глобального і локального планувальника, при якому глобальний модуль періодично будує опорний маршрут на повній карті, а локальний модуль у реальному часі коригує рух з урахуванням нових перешкод і кінематики дрона. Стандартом для такої архітектури сьогодні є навігаційний стек Nav2 у ROS 2 [18][19], який інтегрує більшість розглянутих алгоритмів у формі взаємозамінних модулів. Саме гібридний підхід є базовою архітектурою для подальшої практичної реалізації в межах дипломної роботи.

Висновки до розділу 1

У першому розділі було розглянуто задачу автономної навігації наземних дронів та визначено місце задачі побудови карти глибини й уникнення перешкод у загальній структурі навігаційного стеку. Показано, що наземні безпілотні платформи активно входять у логістичні, інспекційні, агротехнічні та військові задачі, а перехід від телекерування до автономного режиму суттєво підвищує вимоги до системи сприйняття простору. Особливості наземної навігації порівняно з повітряною, зокрема обмежена прохідність і складність сцени з близькими динамічними об'єктами, роблять задачу коректного тривимірного сприйняття критичною для безпеки руху.

Було розглянуто карту глибини як основне проміжне представлення простору між сирими даними сенсора та модулями верхнього рівня. Описано її формальне визначення, зв'язок з картою диспаратності та хмарою точок, відмінність метричних і відносних карт глибини. Проаналізовано особливості

карт глибини для внутрішніх і зовнішніх сцен на прикладах відомих відкритих наборів даних. Зроблено висновок, що якісні характеристики карти глибини суттєво залежать від умов знімання, що впливає на вибір методів подальшої обробки.

Проведено аналіз основних типів сенсорів, які використовуються для отримання інформації про глибину сцени, зокрема монокулярних і стереокамер, кольорових камер з активним вимірюванням глибини та лазерних далекомірів. Розглянуто також додаткові класи обладнання, такі як часо-пролітні камери, ультразвукові датчики й радары. Встановлено, що жоден з типів сенсорів не є універсальним, а у практичних системах часто застосовують комбінацію кількох сенсорів для компенсації слабких сторін окремих класів обладнання.

Окрему увагу приділено класифікації методів побудови карт глибини за двома основними ознаками, типом вхідних даних і характером алгоритму. Виділено родини монокулярних, стереоскопічних, багатоглядових та активних методів, а також поділ на класичні геометричні та сучасні нейромережеві підходи. Розглянуто гібридні методи, які поєднують переваги різних класів алгоритмів і сенсорів.

Розглянуто основні підходи до уникнення перешкод та планування шляху, зокрема алгоритми глобального планування на основі евристичного пошуку та випадкових дерев, методи реактивного уникнення на основі алгоритмів обходу контуру та штучних потенціальних полів, а також локальні планувальники з урахуванням кінематики апарата. Окремо згадано методи на основі навчання з підкріпленням та інші нейромережеві підходи до навігації. Показано, що сучасні навігаційні системи будуються за гібридною архітектурою глобального і локального планувальника, реалізованою у відкритих програмних стеках для робототехніки.

РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ТА ОБҐРУНТУВАННЯ СТРУКТУРИ МОДЕЛІ

2.1 Геометричні основи побудови карт глибини

В основі усіх геометричних методів побудови карт глибини лежить математична модель формування зображення у камері. Найпоширенішою спрощеною моделлю є пінхольна камера, яка описує процес проєкції тривимірної точки сцени на двовимірну площину зображення через єдиний оптичний центр. Усі сучасні алгоритми стереозору, SLAM та більшість методів калібрування ґрунтуються саме на цій моделі.

Нехай точка $P = (x_1, x_2, x_3)^T$ задана у системі координат камери, де вісь x_3 напрямлена вздовж оптичної осі. Проєкція цієї точки на площину зображення відбувається за принципом подібних трикутників. Для отримання координат пікселя у системі координат зображення використовується матриця внутрішніх параметрів камери, що задається у вигляді

$$K = \begin{pmatrix} f_{x_1} & 0 & c_{x_1} \\ 0 & f_{x_2} & c_{x_2} \\ 0 & 0 & 1 \end{pmatrix},$$

де f_{x_1}, f_{x_2} – фокусні відстані у пікселях по осях x_1 та x_2 , а c_{x_1}, c_{x_2} – координати головної точки. Параметри матриці K є індивідуальними для конкретного екземпляра камери і визначаються в процесі калібрування.

Проєкція точки P на піксель (u, v) записується у вигляді

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \frac{1}{x_3} \cdot K \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Формула описує пряму проєкцію тривимірної точки на зображення. Зворотна задача, тобто відновлення глибини x_3 по пікселю (y_1, y_2) , є

принципово неоднозначною для одного зображення, оскільки вздовж проєкційного променя лежить нескінченна множина точок з різними значеннями x_3 . Саме цією неоднозначністю пояснюється потреба у додатковій інформації для побудови карти глибини: другому ракурсі, послідовності кадрів або попередньо навченій моделі.

Загальна схема пінхольної моделі камери наведена на рисунку 2.1.

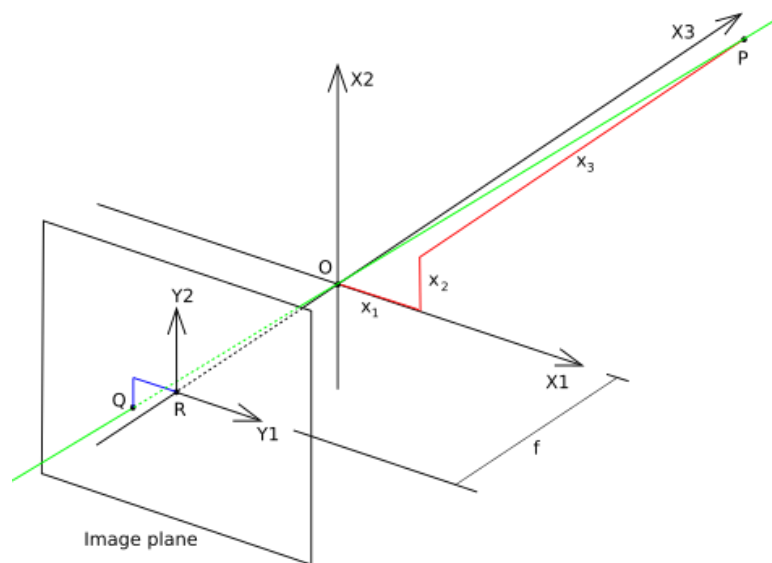


Рисунок 2.1 – Геометрична схема пінхольної моделі камери

Для роботи з кількома камерами одночасно або з рухомою камерою у різні моменти часу потрібно перетворювати координати точок між різними системами відліку. Це перетворення задається матрицею зовнішніх параметрів, яка складається з матриці повороту R розміру 3×3 та вектора зміщення t розміру 3×1 . Перетворення точки з світової системи координат P_w у систему координат камери P_c записується як

$$P_c = R \cdot P_w + t.$$

Внутрішні та зовнішні параметри камери разом утворюють повну геометричну модель формування зображення. Знання цих параметрів є обов'язковою умовою для коректного обчислення глибини у стереозоровому підході та для перетворення карти глибини у хмару точок.

Окремою важливою концепцією є епіпольна геометрія, яка описує співвідношення між двома зображеннями однієї сцени, отриманими з різних ракурсів. Її ключове твердження полягає в тому, що для довільної точки на першому зображенні відповідна точка на другому зображенні лежить на певній прямій, яка називається епіпольною лінією. Це обмеження принципово зменшує область пошуку відповідностей між зображеннями зі всієї площини до однієї лінії і є основою для роботи всіх класичних стереоскопічних алгоритмів. Математично епіпольне обмеження записується через фундаментальну матрицю F як

$$q^{rT} \cdot F \cdot q = 0,$$

де q_1, q' – однорідні координати відповідних точок на першому та другому зображеннях. Фундаментальна матриця визначається взаємним розташуванням камер та їхніми внутрішніми параметрами і обчислюється під час процедури калібрування стереопари.

Калібрування є невід'ємною частиною підготовки будь-якої системи комп'ютерного зору до роботи. Для отримання матриці внутрішніх параметрів зазвичай використовують зйомку об'єкта з відомою геометрією, найчастіше плоскої шахової дошки, у різних положеннях. Найпоширенішим методом калібрування є метод Чжана [10], реалізований у бібліотеці OpenCV [20]. Якісне калібрування є критичною умовою для точності всіх подальших етапів обробки, оскільки помилки в параметрах камери поширюються на всі наступні обчислення глибини та перетворення координат.

2.2 Класичні алгоритми стереозору та SLAM

Класичні методи побудови карт глибини базуються на геометричній моделі камери і не використовують навчених на даних моделей. Найфундаментальнішим з таких підходів є стереозір, у якому глибина обчислюється за різницею положень відповідних точок на двох синхронно знятих зображеннях каліброваної стереопари. Для каліброваної ректифікованої системи зв'язок між диспаратністю та глибиною визначається формулою

$$Z = \frac{f \cdot B}{d},$$

де Z – відстань від камери до точки сцени, f – фокусна відстань камери у пікселях, B – базисна відстань між оптичними центрами камер стереопари, а d – диспаратність відповідних пікселів. Похибка оцінки глибини зростає пропорційно квадрату відстані, а збільшення базису B підвищує точність на дальніх дистанціях, але звужує область одночасної видимості обох камер на близьких.

Ключовою задачею стереозору є знаходження для кожного пікселя лівого зображення відповідного пікселя правого зображення, що зводиться до пошуку диспаратності. Найпростішим класичним підходом є метод блочного зіставлення (Block Matching), у якому навколо кожного пікселя береться блок фіксованого розміру і ведеться пошук найбільш схожого блоку вздовж епіполлярної лінії на другому зображенні. Найпоширенішою мірою подібності є сума абсолютних різниць (Sum of Absolute Differences), яка обчислюється як

$$SAD(d) = \sum_{(i,j) \in W} |I_L(i,j) - I_R(i - d,j)|,$$

де W – вікно зіставлення навколо досліджуваного пікселя, I_L та I_R – інтенсивності лівого та правого зображень, d – значення диспаратності. Альтернативою SAD є сума квадратів різниць, нормалізована крос-кореляція та інші подібні метрики.

Логічним продовженням блочного підходу став алгоритм напівглобального зіставлення [1], запропонований Хайнріхом Хіршмюллером у 2005 році. Він працює як компроміс між простим блочним пошуком і повноцінними глобальними методами. У ньому використовується та сама функція вартості, але до неї додається штраф за різкі стрибки диспаратності між сусідніми пікселями. Оптимізація виконується вздовж 8 або 16 напрямків сканування зображення, після чого результати усереднюються. Завдяки цьому алгоритм значно краще працює на однорідних поверхнях і поблизу країв об'єктів, а швидкість залишається прийнятною для роботи в реальному часі. У бібліотеці OpenCV [20] він доступний у вигляді функції StereoSGBM і часто використовується у робототехнічних проєктах.

Для роботи з однією рухомою камерою класичний стереозір не підходить, бо у нього немає другого кадру для зіставлення. Цю задачу вирішують системи одночасної локалізації та побудови карти. Вони отримують на вхід послідовність кадрів від однієї камери і одночасно обчислюють дві речі: тривимірну структуру сцени та положення самої камери у просторі. Робота таких систем зводиться до пошуку характерних точок на зображенні, їх відстеження між сусідніми кадрами та триангуляції відповідних тривимірних точок. Серед класичних реалізацій найбільш відомі дві: ORB-SLAM [8], яка використовує швидкі бінарні ознаки, та RTAB-Map [9], яка будує щільніші карти і вміє розпізнавати раніше відвідані місця. Для мобільних платформ такі системи зручні тим, що окрім карти глибини вони одразу дають траєкторію руху, яку можна використовувати для локалізації.

Близькою задачею є відновлення структури з руху. Вона теж відновлює тривимірну сцену і положення камер з набору зображень, але працює офлайн.

Системи цього класу беруть готовий набір фото з різних ракурсів і за допомогою методу пакетного коригування одразу оптимізують усі камери та точки разом. Це дозволяє досягати вищої точності, ніж онлайн-системи локалізації та побудови карти, але вимагає більше часу й обчислень. На практиці такий підхід застосовують переважно для тривимірної реконструкції з готових фотознімків, наприклад у фотограмметрії, тоді як для навігації дронів використовують онлайн-системи.

2.3 Нейромережеві методи оцінки глибини

Альтернативою класичним геометричним методам є оцінка глибини за допомогою нейронних мереж. Замість того щоб обчислювати глибину через явні геометричні залежності, модель навчається на великих наборах пар з вхідного зображення та еталонної карти глибини і потім передбачає глибину для нових зображень. Головна перевага такого підходу полягає в можливості працювати з єдиним монокулярним зображенням, для якого класичні методи дають принципово неоднозначний результат. Якісну різницю між класичним і нейромережовим підходами наочно показано на рисунку 2.2.

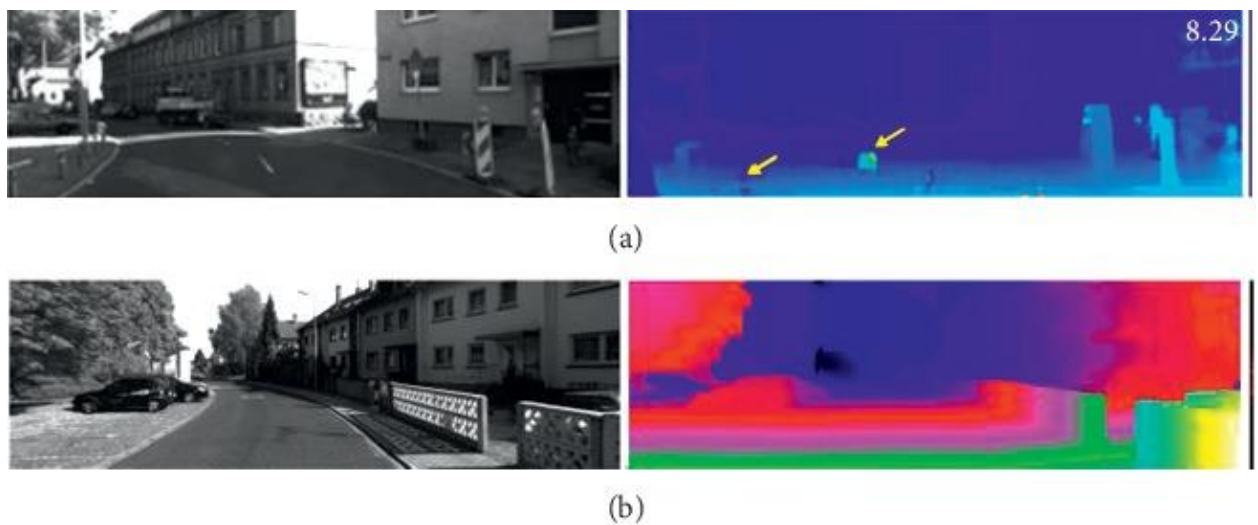


Рисунок 2.2 – Порівняння карт глибини: (а) результат класичного алгоритму напівглобального зіставлення; (б) результат нейромережевого методу

Більшість сучасних моделей оцінки глибини побудовані за схемою енкодер-декодер. Енкодер послідовно зменшує просторову розмірність зображення і виділяє з нього абстрактні ознаки на різних рівнях. Декодер виконує зворотну операцію і поступово відновлює просторову роздільність, формуючи на виході карту глибини того ж розміру, що й вхідне зображення. Між енкодером і декодером часто застосовують пропусчні з'єднання, які передають детальні просторові ознаки повз вузьке місце мережі. У ранніх моделях енкодер будувався на згорткових мережах, а в сучасних на трансформерах для зображень.

За парадигмою навчання моделі поділяються на дві основні групи. Моделі з повним наглядом тренуються на парах зображень з еталонними картами глибини, отриманими від лазерних далекомірів або кольорових камер з активним вимірюванням глибини. Такий підхід дає високу точність, але вимагає великих наборів даних з якісною розміткою. Моделі з самонаглядом не потребують еталонних карт глибини взагалі. Вони навчаються на послідовності кадрів від однієї камери, використовуючи фотометричну

функцію втрат. Модель прогнозує глибину для одного кадру, з її допомогою синтезує сусідній кадр і порівнює його з реальним. Чим точніша глибина, тим менша різниця між синтезованим і реальним кадром. Найвідомішим прикладом такого підходу є модель Monodepth2 [7].

Класичною функцією втрат для тренування з повним наглядом є *scale-invariant loss*, запропонована Девідом Айгеном у 2014 році [2]. Її задача полягає в тому, щоб штрафувати модель не за абсолютне значення глибини, а за її структуру з точністю до загального масштабного коефіцієнта. Формула виглядає так:

$$L = \frac{1}{n} \sum_i d_i^2 - \frac{\lambda}{n^2} (\sum_i d_i)^2,$$

де $d_i = \log \hat{y}^i - \log y^i$ – різниця логарифмів передбаченої та еталонної глибини для i -го пікселя, n – кількість пікселів, λ – коефіцієнт регуляризації. Перший доданок штрафуватиме локальні похибки, а другий пом'якшує штраф за глобальне зміщення масштабу. Завдяки цьому функція стійка до випадків, коли модель передбачає правильну форму сцени, але з невірним абсолютним масштабом.

Серед сучасних монокулярних моделей варто виділити декілька ключових. Модель монокулярної оцінки глибини MiDaS [3], представлена компанією Intel у 2019–2020 роках, стала однією з перших систем, що демонструвала добру якість на сценах будь-якого типу за рахунок тренування на змішаному наборі даних з кількох джерел. Її розвитком став варіант на основі трансформера під назвою DPT [4], який значно покращив роботу з дрібними деталями. Останнім великим кроком стала модель Depth Anything [5], представлена у 2024 році. Вона навчена на 62 мільйонах розмічених і нерозмічених зображень і сьогодні залишається одним з найкращих відкритих рішень для монокулярної оцінки глибини. Її друга версія, Depth Anything V2 [6], покращує результати на тонких структурах і прозорих об'єктах.

Незважаючи на високі результати, у монокулярних нейромережевих моделей залишається кілька суттєвих обмежень. Головне з них стосується невизначеності масштабу. Модель навчається на конкретному розподілі даних і не має фізичної прив'язки до реальних метрів, тому для отримання абсолютної глибини потрібне додаткове калібрування за відомими розмірами об'єктів у сцені, за даними інерціального датчика або за одометрією. Інше обмеження полягає у залежності якості від подібності реальної сцени до навчальних даних. Модель може давати погані результати на нетипових об'єктах, нестандартних умовах освітлення або в доменах, погано представлених у наборах для тренування. Через це в практичних системах нейромережеві моделі часто комбінують з класичними геометричними підходами або з активними сенсорами для контролю якості.

2.4 Обґрунтування вибору моделей для даної роботи

Після того як система сприйняття сформувала карту глибини, цю інформацію потрібно перетворити на представлення, з яким уже працюють алгоритми планування. Найпоширенішим таким представленням є сітка зайнятості. Простір навколо дрона розбивається на комірки фіксованого розміру, кожній з яких присвоюється одне з трьох значень: вільна, зайнята перешкодою або невідома. Сітку зайнятості будують або проєкцією карти глибини на горизонтальну площину, або накопиченням даних з кількох послідовних кадрів. На основі такого представлення задача планування шляху зводиться до пошуку послідовності вільних комірок від поточного положення дрона до цілі.

Класичним алгоритмом глобального планування на сітці зайнятості є A-star [11]. Він розглядає простір комірок як зважений граф і шукає в ньому

найдешевший шлях за допомогою евристичного пошуку. Для кожної комірки n обчислюється сумарна оцінка вартості шляху:

$$f(n) = g(n) + h(n),$$

де $g(n)$ – фактична вартість шляху від початкової точки до комірки n , а $h(n)$ – евристична оцінка вартості від комірки n до цілі. Алгоритм щоразу обирає для розширення комірку з найменшим значенням $f(n)$, що дозволяє знаходити оптимальний шлях значно швидше, ніж простий перебір. Найпоширенішою евристикою для двовимірного простору є манхеттенська або евклідова відстань до цілі. A-star добре працює на статичних картах, але при появі нових перешкод потребує повторного перерахунку всього шляху. Цю проблему вирішує його розширення D-star, яке вміє оновлювати тільки змінені ділянки шляху.

Альтернативою алгоритмам на сітці є методи на основі випадкового вибору. Найвідоміший з них називається RRT [12] і працює в неперервному просторі станів. Алгоритм поступово будує дерево з початкової точки, додаючи на кожному кроці випадково обрану нову вершину і з'єднуючи її з найближчою вже існуючою. Процес триває доти, доки одна з гілок дерева не досягне цільової області. RRT швидко знаходить допустимий шлях навіть у складних середовищах, але не гарантує його оптимальності. Його модифікація RRT-star [13] усуває цей недолік за рахунок додаткової оптимізації під час побудови дерева.

Глобального плану недостатньо для безпечного руху, оскільки на маршруті можуть з'являтися перешкоди, не передбачені картою. Цю задачу вирішує локальний планувальник. Серед них найбільш поширеним для колісних платформ є метод динамічного вікна [14]. Він безпосередньо враховує кінематичні та динамічні обмеження дрона і обирає пару керуючих команд з простору допустимих швидкостей. Кожна можлива пара лінійної швидкості v і кутової швидкості ω оцінюється за функцією вартості:

$$G(v, \omega) = \alpha \cdot heading(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot velocity(v, \omega),$$

де $heading(v, \omega)$ оцінює, наскільки траєкторія від обраних команд напрямлена в бік цілі, $dist(v, \omega)$ – мінімальну відстань до перешкоди вздовж цієї траєкторії, $velocity(v, \omega)$ – лінійну швидкість на ній, а α, β, γ – вагові коефіцієнти, які налаштовуються під конкретну платформу. Дрон обирає ту пару команд, для якої G є максимальною. Такий підхід дає плавний рух з урахуванням реальних обмежень апарата.

Простішим за обчисленнями методом локального планування є метод штучних потенціальних полів. Цільова точка створює навколо себе притягальне поле, а кожна перешкода – відштовхувальне. Дрон рухається вздовж результуючого вектора всіх діючих сил. Перевагою методу є його простота і низька обчислювальна складність. Основним недоліком залишається проблема локальних мінімумів. У певних конфігураціях перешкод сили взаємно компенсуються до того, як дрон досягне цілі, і апарат може застрягти в точці, яка не є кінцевою. Через цю проблему метод штучних потенціальних полів зазвичай не використовують самостійно, а поєднують з іншими алгоритмами планування.

У реальних навігаційних системах розглянуті алгоритми не використовуються поодиночі. Стандартною архітектурою є поєднання глобального планувальника, який працює на повній карті середовища, з локальним планувальником, який враховує поточні дані сенсорів. Глобальний модуль періодично прокладає опорний маршрут до цілі, а локальний у реальному часі коригує рух з урахуванням нових перешкод і фізичних можливостей дрона. Така архітектура реалізована у відкритому навігаційному стеку Nav2 для ROS 2 [18][19], який інтегрує більшість розглянутих алгоритмів у формі взаємозамінних модулів і є основою для практичної реалізації в межах дипломної роботи.

2.5 Метрики оцінювання якості карт глибини

Для порівняння методів побудови карт глибини використовують кілька стандартних метрик, що обчислюються попіксельно між передбаченою та еталонною картами [2]. Кожна характеризує різний аспект якості, тому в публікаціях зазвичай наводять їх набір. Absolute Relative error (AbsRel) обчислюється як

$$AbsRel = \frac{1}{n} \sum_{i=1}^n \frac{|\hat{y}_i - y_i|}{y_i},$$

де $\hat{y}^i$ і y^i – передбачена та еталонна глибина для i -го пікселя, n – кількість пікселів з відомим еталоном. Похибка нормована на реальну відстань, тому однаково чутлива до близьких і далеких об'єктів.

Root Mean Square Error (RMSE) виражається у метрах і визначається формулою:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2}.$$

Квадратична залежність сильніше штрафує великі похибки, тому метрика чутлива до викидів.

Порогові метрики точності показують частку пікселів, для яких відношення передбаченої та еталонної глибини не перевищує заданого порогу:

$$\delta = \frac{1}{n} \left| \left\{ i : \max\left(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right) < 1.25 \right\} \right|.$$

Окрім основного порогу 1.25 використовують також 1.25^2 і 1.25^3 . Для AbsRel і RMSE краще менше значення, для δ – більше.

Метрики навігації стосуються поведінки дрона загалом. Основна – частка успішних запусків, що показує відсоток сценаріїв, у яких дрон досяг цілі без зіткнень. Метрика SPL додатково враховує оптимальність маршруту відносно найкоротшого шляху. Окремо оцінюють кількість зіткнень, час досягнення цілі та плавність траєкторії.

Висновки до розділу 2

У розділі розглянуто математичну та технологічну основу побудови карт глибини й планування шляху для наземних дронів. Описано пінхольну модель камери як базу для геометричних методів, класичні алгоритми стерео і SLAM, а також сучасні нейромережеві підходи до оцінки глибини на основі архітектури енкодер-декодер.

Проаналізовано алгоритми планування шляху та уникнення перешкод, що працюють на основі сітки зайнятості, та метрики, за якими порівнюють методи оцінки глибини. Показано, що практичні системи будуються за гібридною архітектурою глобального і локального планувальника.

Результати розділу формують теоретичну базу для вибору конкретного методу та його практичної реалізації.

РОЗДІЛ 3 РОЗРОБКА МОДЕЛЕЙ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Загальна архітектура системи

Розроблену систему поділено на чотири функціональні частини, кожна з яких виконує окрему задачу і передає свій результат наступній. Такий модульний поділ полегшує налагодження окремих частин, а також дозволяє замінювати реалізацію одного методу без зміни решти програми. Послідовність цих частин і утворює загальний конвеєр обробки даних, який показано на рисунку 3.1.

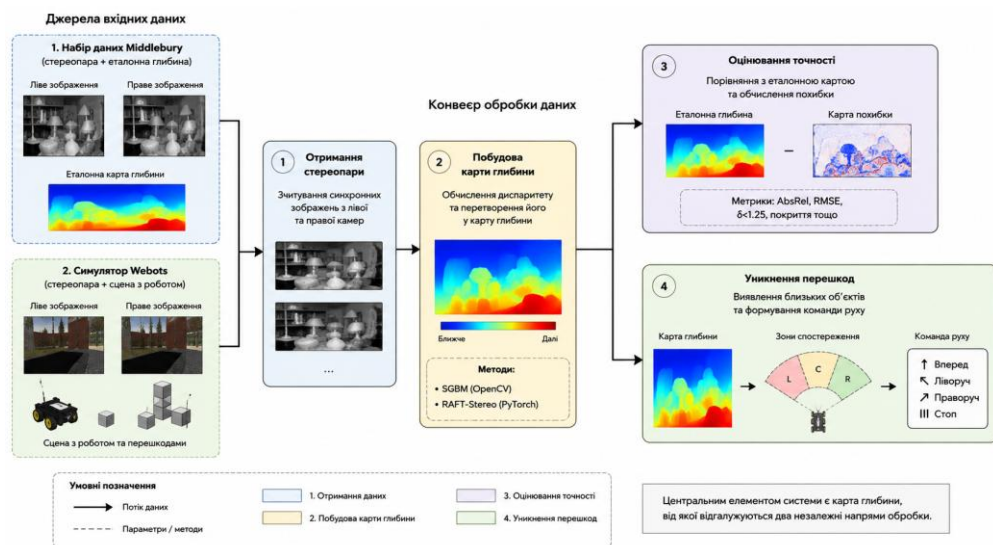


Рисунок 3.1 – Загальна архітектура системи

Перша частина отримує зображення з двох камер, лівої і правої, що разом утворюють стереопару. На основі цієї пари друга частина будує карту глибини. Отриману карту далі використовують одночасно у двох напрямках. Третя частина порівнює її з еталонною картою і обчислює похибку, а четверта

за тією ж картою визначає близькі об'єкти і формує команду руху для їх обходу.

Як видно з наведеної послідовності, центральним елементом системи виступає саме карта глибини, оскільки всі подальші частини працюють з нею, а не з вихідними зображеннями. Від неї відгалужуються два незалежні напрями обробки, які відповідають двом складовим поставленої задачі. Перший напрям полягає в оцінюванні точності відновлення глибини шляхом порівняння з еталоном, а другий у практичному застосуванні цієї глибини для уникнення перешкод.

Вхідні зображення надходять із двох джерел, які доповнюють одне одного. Першим джерелом є набір даних Middlebury, який містить готові стереопари з еталонними картами глибини [21]. Він дає змогу об'єктивно порівняти методи між собою на однакових вхідних даних з відомою істинною глибиною. Другим джерелом є симулятор Webots, у якому змодельовано сцену з роботом і перешкодами [22]. Він надає пару камер для побудови глибини і керовану сцену з рухомим роботом, потрібну для перевірки уникнення перешкод. Ці два джерела відповідають двом гілкам роботи. Гілка набору даних слугує для кількісного порівняння точності методів, а гілка симулятора для перевірки практичного застосування глибини в задачі руху.

Через відмінність задач параметри камер у двох гілках різні. У гілці набору даних геометричні параметри стереопари задано файлами калібрування Middlebury, окремими для кожної сцени. У гілці симулятора параметри камер обрано так, щоб карта глибини будувалась у реальному часі під час руху робота.

Відповідно до описаної структури програмну реалізацію системи виконано мовою Python. Класичний метод побудови глибини реалізовано засобами бібліотеки OpenCV [20], нейромережевий метод за допомогою бібліотеки PyTorch [23], а моделювання сцени виконано в середовищі Webots [22].

3.2 Моделювання сцени та стереопари в середовищі Webots

Для моделювання робота і навколишнього середовища обрано симулятор Webots [22]. Це відкрите середовище, призначене саме для робототехнічного моделювання, тому воно вже містить готові моделі роботів, набір сенсорів і фізичний рушій. На відміну від універсальних ігрових середовищ, у Webots можна під'єднати сенсори безпосередньо до моделі робота і керувати ними програмно.

Саму сцену змодельовано у вигляді прямокутної арени з підлогою і стінками. Усередині арени розміщено кілька перешкод простої форми на різній відстані від робота, щоб карта глибини містила об'єкти на різних дистанціях. Перешкодам надано вираженої текстури, оскільки стереозіставлення спирається на пошук однакових ділянок у двох зображеннях і на однотонних поверхнях відповідностей не знаходить. Як мобільну платформу використано колісного робота Pioneer 3-DX, який входить до стандартного набору моделей Webots [22]. Робот має диференційний привід, тобто керується незалежними швидкостями лівого і правого коліс.

На робота встановлено стереопару з двох камер. Камери розташовано паралельно і на одній горизонтальній лінії, зміщеними одна відносно одної на 0,10 метра вздовж горизонтальної осі. Таке розташування відповідає геометрії ректифікованої стереопари і є потрібною умовою для подальшого зіставлення зображень. Камери мають роздільну здатність 320 на 240 пікселів і горизонтальний кут огляду 1,5 радіана. У середовищі Webots камери є ідеальними, тому зображення не містять шуму, викривлень об'єктива і розфокусу, а епіполярні лінії точно горизонтальні.

Керування роботом реалізовано окремим контролером, написаним мовою Python. На кожному кроці контролер зчитує зображення з обох камер, будує за ними карту глибини, визначає за нею близькі перешкоди і задає

швидкості коліс. Цей контролер працює в режимі реального часу під час руху, на відміну від обробки набору даних, яку виконують окремо й однократно.

3.3 Калібрування стереопари та геометричні перетворення

Щоб перейти від зображень до реальних відстаней, потрібно знати геометричні параметри камер. В основі цього переходу лежить модель камери-обскури, відома також як *pinhole*-модель. За нею кожна точка тривимірного простору проектується на площину зображення через єдиний оптичний центр. Камеру в цій моделі описують її внутрішні параметри, насамперед фокусна відстань і координати головної точки [10].

Фокусну відстань визначають по-різному для двох гілок роботи. Для сцен набору даних Middlebury її задано у файлі калібрування [21], що супроводжує кожну сцену, разом з базою стереопари і зсувом головних точок. Для камер симулятора Webots окреме калібрування не потрібне, бо камери ідеальні і не мають викривлень об'єктива. У цьому разі фокусну відстань обчислюють безпосередньо з кута огляду і роздільної здатності за формулою

$$f = W / (2 \cdot \tan(FOV / 2)),$$

де W позначає ширину зображення в пікселях, а FOV горизонтальний кут огляду.

Стереопара складається з двох однакових камер, зміщених одна відносно одної на відстань B уздовж горизонтальної осі. Цю відстань називають базою стереопари. Оскільки камери орієнтовано паралельно і розміщено на одній лінії, їхні епіполлярні лінії є горизонтальними, тому відповідні точки на лівому і правому зображеннях завжди лежать в одному рядку. Така конфігурація відповідає ректифікованій стереопарі і дозволяє

шукати відповідні точки лише вздовж рядків, що суттєво спрощує зіставлення. У моделюванні Webots, як і в наборі даних Middlebury, зображення вже є ректифікованими, тому окреме усунення викривлень не виконують.

Маючи фокусну відстань і базу, глибину точки Z обчислюють за її диспаратністю d , тобто за різницею горизонтальних координат проекцій цієї точки на лівому і правому зображеннях. Залежність має вигляд

$$Z = f \cdot B / d.$$

З формули видно, що глибина обернено пропорційна диспаратності. Чим ближчий об'єкт до камер, тим більший зсув його проекцій між зображеннями, і навпаки, віддалені об'єкти мають малу диспаратність. Для набору даних Middlebury у цю залежність додають ще один доданок, який враховує зсув головних точок лівої та правої камер і позначається $doffs$. Тоді формула набуває вигляду

$$Z = f \cdot B / (d + doffs).$$

Значення фокусної відстані, бази і зсуву головних точок для сцен Middlebury задано у файлі калібрування. Для камер симулятора зсув головних точок дорівнює нулю. База стереопари в симуляторі відома заздалегідь із взаємного розташування камер і дорівнює 0,10 метра.

3.4 Побудова карти глибини зі стереопари

Побудова карти глибини полягає в тому, щоб для кожного пікселя лівого зображення знайти відповідний йому піксель правого зображення і обчислити диспаратність між ними. Знайдену диспаратність потім переводять у глибину.

Основну складність у цій задачі становить саме пошук відповідних пікселів, бо від його точності залежить якість усієї карти глибини.

У роботі розглянуто два різні підходи до розв'язання цієї задачі. Першим є класичний алгоритм напівглобального блокового зіставлення SGBM [1], який знаходить відповідності безпосередньо за вмістом зображень і не потребує попереднього навчання. Другим є нейромережевий метод RAFT-Stereo [24], що навчається на наборах даних і належить до сучасних методів оцінювання глибини. Перший підхід відображає традиційний спосіб розв'язання задачі, а другий показує можливості сучасних навчених моделей.

Обидва методи дають на виході карту диспаратності, тому глибину з неї обчислюють однаково, за тими самими формулами. Завдяки цьому методи можна порівняти між собою на однакових вхідних даних і за однаковими метриками якості. Класичний SGBM узято як прозорий і добре вивчений еталон, а RAFT-Stereo як представник нейромережевих методів, щоб з'ясувати, наскільки сучасний підхід точніший за класичний. Безпосереднє порівняння результатів обох методів наведено в розділі 4.

3.4.1 Класичний метод SGBM

Першим із розглянутих методів є напівглобальне блокове зіставлення, відоме під назвою Semi-Global Block Matching, або скорочено SGBM. Цей метод знаходить диспаратність, спираючись лише на вміст зображень, і не потребує попереднього навчання на даних. В його основі лежить ідея зіставлення невеликих ділянок лівого і правого зображень з подальшим узгодженням результату по всьому зображенню.

Робота методу починається з обчислення вартості зіставлення. Для кожного пікселя лівого зображення і для кожного можливого значення

диспаратності алгоритм оцінює, наскільки добре відповідна ділянка лівого зображення збігається з ділянкою правого, зміщеною на цю диспаратність. Чим кращий збіг, тим менша вартість. Якщо орієнтуватися лише на неї, карта диспаратності виходить шумною, бо окремі пікселі легко зіставляються помилково.

Щоб уникнути таких помилок, метод додає до вартості штрафи за різку зміну диспаратності між сусідніми пікселями. Невелику зміну, на один рівень диспаратності, штрафують величиною $P1$, а більшу зміну штрафують більшою величиною $P2$. Завдяки цьому сусідні пікселі схиляються до близьких значень диспаратності, і карта стає гладкою там, де поверхня насправді неперервна, але зберігає різкі переходи на межах об'єктів.

Узгодження цих штрафів по всьому зображенню одразу було б надто складним з обчислювального погляду. Тому метод виконує оптимізацію не глобально, а вздовж кількох напрямків, що проходять через кожен піксель, після чого підсумовує отримані вартості. Саме такий підхід і дав методу назву напівглобального, оскільки він наближає глобальну оптимізацію, але коштує значно дешевше.

У роботі метод реалізовано засобами бібліотеки OpenCV за допомогою класу StereoSGBM [20]. Найважливішим його параметром є діапазон пошуку диспаратності, який задають кратним шістнадцяти і узгоджують з максимальною очікуваною диспаратністю сцени. Для сцен набору даних Middlebury цей діапазон визначають із файлу калібрування. Розмір блоку зіставлення обрано рівним п'яти пікселям, а штрафи $P1$ і $P2$ задано за рекомендованими співвідношеннями, пропорційними до квадрата розміру блоку. Додатково використано параметри відсіювання ненадійних зіставлень, які прибирають дрібні помилкові ділянки і перевіряють однозначність знайденої відповідності.

Результатом роботи методу є карта диспаратності. Для частини пікселів надійного зіставлення знайти не вдається. Це стосується ділянок, видимих

лише однією камерою через перекриття об'єктами, однотонних поверхонь без вираженої текстури, а також країв зображення. Такі пікселі позначають як невизначені і не враховують під час оцінювання якості.

3.4.2 Нейромережевий метод RAFT-Stereo

Другим розглянутим методом є RAFT-Stereo, запропонований у роботі Lipson, Teed і Deng у 2021 році [24]. Це нейромережевий метод побудови карти диспаратності, що належить до сучасних навчених підходів. Принципова відмінність від SGBM полягає в джерелі знань про відповідність точок. Класичний метод спирається на заздалегідь задані правила зіставлення блоків, тоді як RAFT-Stereo навчається знаходити відповідності на великій кількості стереозображень із відомою диспаратністю і застосовує набутий досвід до нових сцен. У цій роботі метод не реалізовано заново, а використано офіційну реалізацію його авторів разом із попередньо натренованими вагами.

Робота методу починається з того, що згорткова мережа обробляє ліве і праве зображення та переводить їх із набору окремих пікселів у набір ознак, які описують вміст сцени стійкіше, ніж яскравість окремих точок. На основі цих ознак для кожної точки лівого зображення оцінюють її схожість з можливими відповідниками на правому. Сукупність таких оцінок утворює міру близькості, якою метод далі користується замість прямого порівняння яскравостей, як це робить SGBM.

Особливістю RAFT-Stereo є спосіб, у який він знаходить остаточну диспаратність. Замість того щоб одразу обрати найкращу відповідність, метод будує початкову грубу оцінку і потім багаторазово її уточнює. На кожному кроці він звіряє поточну оцінку з мірою близькості й вносить у неї поправку, поступово наближаючись до узгодженого результату. Уточнення виконують

на кількох рівнях деталізації, тому метод враховує і загальну будову сцени, і дрібні деталі на межах об'єктів. Саме цей повторюваний процес уточнення відображено в назві методу.

Важливою властивістю методу є щільність результату. Оскільки оцінка послідовно поширюється на весь кадр, RAFT-Stereo визначає диспаратність для кожного пікселя зображення і не лишає невизначених ділянок. Це принципово відрізняє його від SGBM, який не дає відповіді в зонах оклюзій, на однотонних поверхнях і поблизу країв зображення. Щільна карта зручніша для подальшого використання, бо не потребує окремого заповнення пропусків.

У роботі використано ваги RAFT-Stereo, попередньо натреновані лише на синтетичному наборі Scene Flow [25], без донавчання на реальних даних. Такий вибір не випадковий. Оскільки класичний SGBM не використовує навчання взагалі, для порівняння нейромережу теж застосовано на сценах Middlebury, які не брали участі в її навчанні. Метод застосовано до тих самих підготовлених сцен і з тією самою роздільністю, що й SGBM, та оцінено тими самими метриками.

3.5 Виявлення перешкод за картою глибини

Карта глибини задає відстань для кожного пікселя кадру, але для керування роботом такий обсяг даних надлишковий. Роботу потрібно знати лише, чи є попереду близька перешкода і з якого боку вона ближча. Тому карту глибини зводять до трьох чисел, які описують відстань до найближчих перешкод ліворуч, по центру і праворуч. Усі викладки цього підрозділу

стосуються гілки симулятора, де карту глибини будують у реальному часі під час руху.

Карту глибини будують не на повному кадрі, а на зменшеному вдвічі задля швидкості. Камери дають кадр розміром 320 на 240 пікселів, а зіставлення виконують на кадрі 160 на 120.

Для аналізу беруть нижню половину карти, де видно підлогу перед роботом і перешкоди на шляху. Це 60 нижніх рядків зменшеного кадру. Зліва відкидають смугу завширшки $d_{max} = 32$ пікселі, бо в ній праве зображення не має відповідної ділянки і глибину там не визначено в принципі. Решту області завширшки 128 пікселів ділять на три рівні вертикальні зони приблизно по 43 пікселі, які відповідають напрямкам ліворуч, прямо і праворуч.

У кожній зоні багато значень глибини, і з них дістають одне число, що відповідає відстані до найближчої перешкоди. Мінімум для цього не годиться, бо він надто чутливий до поодиноких помилкових пікселів. Замість мінімуму беруть десятий перцентиль глибини по всіх валідних пікселях зони:

$$d_z = P_{10}(\{Z_i : \text{піксель } i \text{ зони валідний}\}).$$

Десятый перцентиль відповідає ближній частині зони, але стійкий до рідкісних викидів. Три отримані відстані d_L , d_C , d_R і є результатом виявлення перешкод.

Оскільки, стереозіставлення дає глибину не для всіх пікселів, невалідні пікселі виникають у двох випадках. Перший випадок це об'єкт, ближчий за мінімальну вимірювану відстань, яка задається діапазоном пошуку диспаратності:

$$Z_{min} = f_s \cdot B / d_{max} \quad (3.6).$$

Для $f_s \approx 85,9$ пікселя, бази $B = 0,10$ м і $d_{\max} = 32$ ця відстань дорівнює $Z_{\min} \approx 0,27$ м. Об'єкт, ближчий за $Z_{\min}$, дає диспаратність поза діапазоном, і його пікселі лишаються невизначеними. Другий випадок це поверхня без вираженої текстури. Однотонна стіна не дає ознак для зіставлення, тому її пікселі невалідні на будь-якій відстані. Через це невалідний піксель неоднозначний і може означати як перешкоду впритул, так і вільну далечінь зі слабкою текстурою.

Щоб ця неоднозначність не призводила до хибних рішень, валідність перевіряють лише в нижній частині зони. Беруть нижні сорок відсотків зони, тобто найближчу до робота смугу робочої області, і рахують у них частку валідних пікселів. Зону вважають заблокованою, коли ця частка менша за поріг τ :

$$N_{\text{valid}}(\text{near}) / N(\text{near}) < \tau, \tau = 0,25.$$

Невалідність у верхній частині зони означає віддалену або слаботекстуровану ділянку і зону не блокує. Таке правило знімає обидва помилкові тлумачення. Близька однотонна стіна перед роботом не сприймається як вільний простір, а далекий слаботекстурований фон не сприймається як стіна.

3.6 Алгоритм уникнення перешкод

Алгоритм уникнення отримує три відстані d_L , d_C , d_R і перетворює їх на швидкості коліс. Він побудований як скінченний автомат із трьома станами: рух уперед, поворот і задній хід. Стан задає характер руху, а відстані

в зонах задають переходи між станами. Робот реагує лише на поточний кадр і не будує карти всього середовища.

Переходи між станами визначають через чотири порогові відстані. Поріг $DANGER = 0,7$ м відповідає небезпечній близькості перешкоди попереду, $SAFE = 1,4$ м відповідає достатньо вільному простору для виходу з повороту, $SIDE_CRITICAL = 0,4$ м відповідає критичній близькості збоку, а $CRITICAL = 0,3$ м відповідає блокуванню з усіх боків. Пороги входу і виходу різні, бо $SAFE$ більший за $DANGER$. Ця різниця створює гістерезис і не дає роботу перемикатися між рухом і поворотом на межі одного порога.

У стані руху вперед робот неперервно підвертає до вільнішого боку. Для цього обчислюють зміщення керування за різницею бічних відстаней:

$$bias = (d_R - d_L) / D_{steer},$$

де $D_{steer} = 1,5$ м, а самі d_L і d_R перед обчисленням обмежують зверху значенням D_{steer} , тому $bias$ лежить у межах від мінус одиниці до одиниці. За цим зміщенням задають швидкості лівого і правого коліс симетрично відносно крейсерської швидкості:

$$v_L = CRUISE \cdot (1 + k \cdot bias), \quad v_R = CRUISE \cdot (1 - k \cdot bias),$$

де $CRUISE = 0,4 \cdot v_{max}$ – крейсерська швидкість, $v_{max} = 12,3$ рад/с – максимальна кутова швидкість колеса, $k = 0,6$ – коефіцієнт підруління. Швидкості обмежують діапазоном від нуля до v_{max} . Такий спосіб керування є спрощеним методом потенціальних полів [26]. У вузьких проходах він центрує робота, а від стін відводить плавно.

Перехід до повороту відбувається у двох випадках. Перший настає, коли перешкода попереду ближча за поріг, тобто $d_C < DANGER$. Тоді робот повертає до того боку, де відстань більша. Другий настає, коли якийсь бік ближчий за $SIDE_CRITICAL$, навіть якщо попереду вільно. Цей випадок закриває підхід до стіни під гострим кутом, коли центральна зона дивиться

вздовж стіни і здається вільною. У цій ситуації робот розвертається від ближчого боку.

У стані повороту робот обертається на місці зі швидкостями коліс

$$v_L = -s \cdot 0,3 \cdot CRUISE, v_R = s \cdot 0,3 \cdot CRUISE,$$

де s дорівнює $+1$ для повороту ліворуч і -1 для повороту праворуч. Напрямок обирають на вході в стан і не змінюють до виходу. Вихід відбувається, коли $d_C > SAFE$ і обидва боки більші за $SIDE_CRITICAL$. Якщо робот лишається в повороті понад $SAFE_RELAX_TICKS = 15$ рішень, умову послаблюють до $d_C > DANGER$, щоб він міг рушити через будь-який прохідний просвіт. Коли поворот триває довше за $TURN_TIMEOUT = 50$ рішень, спрацьовує задній хід.

Стан заднього ходу вмикається за тайм-аутом повороту, а також коли всі три зони ближчі за $CRITICAL$, що означає глухий кут. Робот їде назад зі швидкістю $-0,3 \cdot CRUISE$ протягом $BACKUP_TICKS = 12$ рішень, після чого знову переходить до повороту.

Окремо передбачено вихід із локальних пасток. У просторі між кількома перешкодами реактивний алгоритм може зациклитися, бо без пам'яті в однаковій обстановці він щоразу приймає однакове рішення. Щоб розірвати цикл, рахують кількість послідовних циклів повороту і заднього ходу без виходу вперед. Кожен наступний задній хід роблять довшим, до трьох разів, а після $ESCAPE_AFTER = 2$ невдалих циклів напрямком повороту обирають випадково. Випадковий вибір порушує повторюваність рішень, тому вічне зациклення стає неможливим, хоча час виходу стає випадковим.

Рішення приймають не на кожному кроці симуляції, а раз на кілька кроків. Період позначимо Δt , він дорівнює добутку кількості пропущених кроків на тривалість одного кроку симуляції. Між рішеннями мотори утримують останню задану швидкість. Такий період обрано тому, що контролер у симуляторі працює синхронно і симулятор очікує завершення

обчислення карти глибини, перш ніж продовжити. Через це час побудови карти прямо обмежує частоту рішень.

Лінійну швидкість робота визначають з кутової швидкості коліс. Колесо Pioneer 3-DX має радіус приблизно 0,0975 м, тому крейсерській кутовій швидкості відповідає лінійна швидкість $v \approx 0,48$ м/с. Пороги узгоджено з цією швидкістю і періодом рішень за умовою

$$DANGER \geq Z_{min} + 2 \cdot v \cdot \Delta t + \text{запас.}$$

Висновки до розділу 3

У третьому розділі розроблено структуру системи побудови карти глибини та уникнення перешкод на основі стереозору. Запропоновано модульну архітектуру, що об'єднує отримання стереозображень, відновлення глибини, оцінювання точності та формування керуючих команд для мобільного робота.

Створено модель робота та навколишнього середовища в середовищі Webots і сформовано стереопару камер із заданими геометричними параметрами. Розглянуто математичні основи визначення глибини за диспаратністю та особливості калібрування стереосистеми для набору даних Middlebury і моделі в симуляторі.

Для побудови карти глибини реалізовано два підходи: класичний метод напівглобального блокового зіставлення SGBM та нейромережевий метод RAFT-Stereo. Обидва методи забезпечують отримання карти диспаратності та дозволяють виконувати подальше порівняння за однаковими критеріями якості.

Розроблено алгоритм виявлення перешкод, який перетворює карту глибини на компактне представлення простору у вигляді трьох зон спостереження. На основі отриманої інформації реалізовано реактивний алгоритм уникнення перешкод у вигляді скінченного автомата, що забезпечує вибір режимів руху, повороту та заднього ходу, а також передбачає механізми виходу з локальних пасток.

Отримані результати створюють основу для подальшого експериментального дослідження та кількісного порівняння розглянутих методів побудови карти глибини.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1 Методика проведення експериментів

Експериментальне дослідження переслідує дві мети. Перша полягає в оцінюванні точності побудованих карт глибини і порівнянні між собою двох обраних методів – класичного SGBM і нейромережевого RAFT-Stereo. Друга стосується перевірки роботи алгоритму уникнення перешкод, що спирається на побудовану карту глибини. Цей підрозділ описує методику першої, основної частини дослідження, оскільки саме вона визначає вибір методу побудови глибини для всієї системи. Методику дослідження уникнення перешкод винесено в підрозділ 4.4 і розглянуто там разом із відповідними результатами.

Для оцінювання точності обрано набір даних Middlebury 2021. Він містить стереопари реальних сцен, для кожної з яких відомі параметри калібрування і еталонна карта диспаратності, отримана незалежним високоточним вимірюванням. Вибір саме цього набору зумовлений кількома міркуваннями. По-перше, його сцени вже ректифіковані й супроводжуються калібруванням, що повністю відповідає геометричним умовам, прийнятим у розділах 2 і 3, тому жодних додаткових перетворень над вхідними даними не потрібно. По-друге, наявність незалежного еталона дає змогу оцінювати похибку об'єктивно, а не наближено. По-третє, сцени набору мають різну складність і містять, зокрема, однотонні та відбивні поверхні, тому придатні для перевірки поведінки методів не лише в простих умовах, а й там, де стереозіставлення традиційно помиляється. Усього в експериментах використано 24 сцени набору.

Щоб порівняння методів було коректним, обидва застосовано в однакових умовах. Кожен метод опрацьовує ті самі сцени, зменшені до

однакової роздільності завширшки 1500 пікселів, з тим самим калібруванням і тим самим еталоном. Метод RAFT-Stereo застосовано в режимі узагальнення – з вагами, натренованими лише на синтетичному наборі Scene Flow, без донавчання на сценах Middlebury. Завдяки цьому жоден із методів не має переваги від попереднього знайомства з тестовими сценами: класичний SGBM навчання не використовує взагалі, а нейромережа працює на даних, яких не бачила під час тренування. Карти глибини обох методів оцінено одним і тим самим програмним кодом за однаковими формулами.

Для кількісного оцінювання використано такі метрики як: середню відносну похибку глибини AbsRel, середньоквадратичну похибку RMSE, частку пікселів у межах порогу $\delta < 1,25$ і покриття, тобто частку пікселів, для яких метод узагалі визначив глибину. Перші три метрики характеризують точність відновленої глибини, а покриття – її повноту, що особливо важливо для порівняння розрідженого SGBM зі щільним RAFT-Stereo. Перед обчисленням RMSE глибину обмежують зверху значенням, яке відповідає максимальній глибині сцени з невеликим запасом. Це потрібно, щоб поодинокі грубо помилкові пікселі, для яких метод дає майже нульову диспаратність і через це надмірно велику глибину, не спотворювали підсумкову похибку. Решту метрик обчислюють без обмежень.

Експеримент проводять окремо для кожної сцени. Спершу обраним методом будують карту диспаратності за стереопарою, потім переводять її в метричну глибину і порівнюють з еталоном лише на тих пікселях, де визначено і прогноз методу, і еталонне значення. Отримані для кожної сцени метрики зводять у спільну таблицю, а для узагальненої оцінки обчислюють їхнє середнє і медіану по всіх 24 сценах. Поряд із середнім подають саме медіану, оскільки вона стійкіша до окремих складних сцен і краще відображає типову якість методу, а зіставлення середнього з медіаною додатково показує розкид якості між сценами різної складності.

4.2 Формування еталонних карт глибини

Об'єктивне оцінювання точності можливе лише за наявності еталонної карти глибини, з якою порівнюють результат методу. Для реальних сцен таким еталоном слугують карти набору Middlebury 2021, у якому справжню глибину виміряно незалежно від стереозіставлення. Вимір виконано методом структурованого підсвічування: на сцену проєктують відомий світловий візерунок і за його спотворенням на поверхнях відновлюють їхню форму. Оскільки цей спосіб не спирається на пошук відповідних точок між двома зображеннями, він не повторює похибок методів SGBM і RAFT-Stereo, які перевіряються. Саме це робить такий еталон придатним як незалежна опорна правда: якби його будували тим самим стереозіставленням, він успадкував би ті самі помилки, і оцінювання втратило б сенс.

Еталонні дані задано у вигляді карти диспаратності. Щоб порівнювати їх із результатами методів, кожен еталонну карту переведено в метричну глибину за параметрами калібрування відповідної сцени, що й карти, побудовані розглянутими методами. Завдяки цьому еталон і прогнози методів виражено в однакових одиницях і в одному масштабі, що дає змогу зіставляти їх поточно.

Еталон покриває сцену не повністю. Частина пікселів у ньому позначено як невизначені – насамперед ділянки, видимі лише однією камерою, та поверхні, від яких світловий візерунок відбився погано. Під час оцінювання їх вилучено, і всі метрики обчислено лише на пікселях, для яких справжня глибина відома. Такий підхід запобігає викривленню результатів, бо метод не штрафують і не заохочують там, де порівнювати немає з чим.

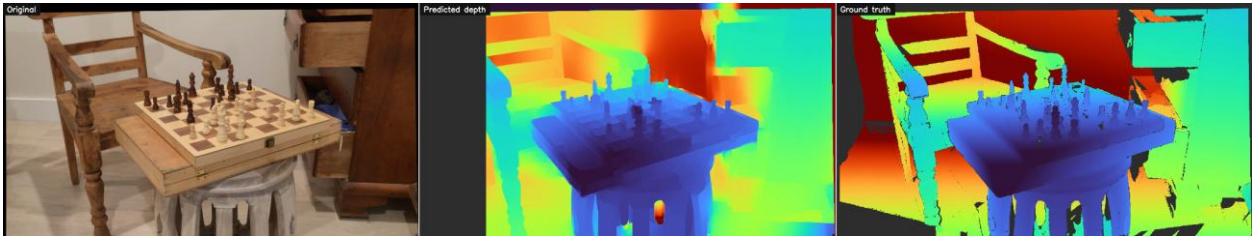


Рисунок 4.1 – приклад еталонної карти глибини сцени Middlebury: ліворуч зображення сцени, всередині прогнозована карта глибини, праворуч відповідна еталонна карта глибини.

4.3 Порівняння якості карт глибини методами SGBM та RAFT-Stereo

Обидва методи застосовано до тих самих 24 сцен набору Middlebury 2021 в однакових умовах, описаних у підрозділах 4.1 і 4.2, і оцінено тими самими метриками. RAFT-Stereo працював у режимі узагальнення, тобто з вагами, натренованими лише на синтетичному наборі Scene Flow, і сцен Middlebury під час навчання не бачив. Результати поточкового порівняння з еталоном по кожній сцені, а також їхні середнє і медіану наведено в таблиці 4.1.

Таблиця 4.1 – Порівняння точності SGBM і RAFT-Stereo на сценах Middlebury 2021

<i>Сцена</i>	<i>AbsRel</i> (SGBM / RAFT)	<i>RMSE, м</i> (SGBM / RAFT)	$\delta < 1,25, \%$ (SGBM / RAFT)	<i>Покриття, %</i> (SGBM / RAFT)
artroom1	0.068 / 0.011	0.94 / 0.68	97.3 / 99.2	80 / 100
artroom2	0.028 / 0.012	0.81 / 0.44	97.6 / 99.4	81 / 100
bandsaw 1	0.118 / 0.006	0.64 / 0.20	97.3 / 99.9	82 / 100
bandsaw 2	0.091 / 0.012	0.38 / 0.15	97.9 / 100.0	66 / 100
chess1	0.363 / 0.018	0.18 / 0.38	94.3 / 98.5	65 / 100
chess2	0.140 / 0.013	0.20 / 0.07	91.8 / 99.3	64 / 100
chess3	0.560 / 0.019	0.18 / 0.15	93.3 / 97.5	72 / 100
curule1	0.147 / 0.014	0.18 / 0.07	95.2 / 99.4	77 / 100
curule2	0.127 / 0.017	0.17 / 0.12	95.5 / 98.6	76 / 100
curule3	0.045 / 0.016	0.13 / 0.09	96.9 / 98.8	83 / 100

Кінець таблиці 4.1

<i>Сцена</i>	<i>AbsRel</i> (<i>SGBM</i> / <i>RAFT</i>)	<i>RMSE</i> , <i>m</i> (<i>SGBM</i> / <i>RAFT</i>)	$\delta < 1,25$, % (<i>SGBM</i> / <i>RAFT</i>)	<i>Покриття</i> , % (<i>SGBM</i> / <i>RAFT</i>)
ladder1	0.063 / 0.019	1.27 / 0.47	93.8 / 99.6	80 / 100
ladder2	0.065 / 0.022	1.11 / 0.52	94.1 / 98.1	80 / 100
octogons 1	0.191 / 0.016	1.34 / 0.41	92.7 / 99.4	80 / 100
octogons 2	0.252 / 0.011	1.11 / 0.31	91.6 / 99.5	84 / 100
pendulu m1	0.143 / 0.009	2.20 / 0.48	96.4 / 99.5	76 / 100
pendulu m2	0.898 / 0.012	5.69 / 0.41	87.2 / 99.6	66 / 100
podium1	0.401 / 0.023	2.64 / 0.65	81.6 / 99.2	49 / 100
skates1	0.022 / 0.012	0.14 / 0.06	97.5 / 99.6	65 / 100
skates2	0.031 / 0.014	0.11 / 0.06	99.2 / 99.8	82 / 100
skiboats 3	0.069 / 0.012	0.13 / 0.07	97.5 / 99.3	80 / 100

За всіма чотирма метриками RAFT-Stereo переважає SGBM, причому перевага стабільна: за відносною похибкою AbsRel і за покриттям нейромережевий метод виявився кращим на всіх 24 сценах без винятку.

Середня відносна похибка глибини в RAFT-Stereo становить 0,016 проти 0,185 у SGBM, а за медіаною – 0,014 проти 0,122, тобто менша приблизно на порядок. Середньоквадратична похибка також менша: медіана 0,26 м проти 0,51 м. Частка пікселів у межах порогу $\delta < 1,25$ у RAFT-Stereo сягає в середньому 98,8% проти 94,4% у SGBM.

Найпомітніша відмінність стосується покриття. RAFT-Stereo визначає глибину для всіх пікселів кадру – покриття 100% на кожній сцені, – тоді як SGBM покриває в середньому лише 75%, а подекуди опускається до 49% (сцена *podium1*). Це прямо підтверджує різницю, закладену в самій природі методів і описану в підрозділі 3.4: класичне блокове зіставлення лишає невизначеними ділянки оклюзій, однотонні поверхні та краї кадру, а RAFT-Stereo за побудовою дає щільну карту без пропусків.

Перевага RAFT-Stereo найбільша саме на складних сценах. Для SGBM середнє значення *AbsRel* помітно перевищує медіану (0,185 проти 0,122): кілька важких сцен різко псують середній показник. Це сцени з відбивними та слаботекстурованими поверхнями – *pendulum2* з похибкою *AbsRel* 0,90, *chess3* з 0,56, *podium1* з 0,40, *chess1* з 0,36, – де пошук відповідних блоків систематично помиляється. На цих самих сценах RAFT-Stereo лишається біля свого найкращого рівня з похибкою 0,01–0,03 і повним покриттям, тому розкид його якості між сценами малий: середнє 0,016 при медіані 0,014. Отже, точність SGBM сильно залежить від складності сцени, тоді як RAFT-Stereo тримає якість стабільною.

Оскільки RAFT-Stereo застосовано в режимі узагальнення і він не навчався на наборі *Middlebury*, наведена перевага є властивістю самого методу, а не наслідком підлаштування під цей набір. Водночас розглянуті метрики характеризують лише точність відновлення глибини. Практичний вибір методу залежить ще й від обчислювальної вартості: RAFT-Stereo потребує значно більших обчислювальних ресурсів і графічного прискорювача, тоді як SGBM працює помітно швидше.

4.4 Дослідження роботи алгоритму уникнення перешкод

Працездатність алгоритму уникнення виявляється не в точності окремих відстаней, а в характері руху робота серед перешкод, тому дослідження цього алгоритму має якісний характер. За роботом спостерігали в кількох характерних ситуаціях, кожна з яких навантажує окрему частину алгоритму керування. Метою такого спостереження було встановити, чи правильно спрацьовують передбачені переходи між станами і чи відповідає поведінка робота закладеній логіці. Усі прогони виконано на сцені, змодельованій у середовищі Webots.

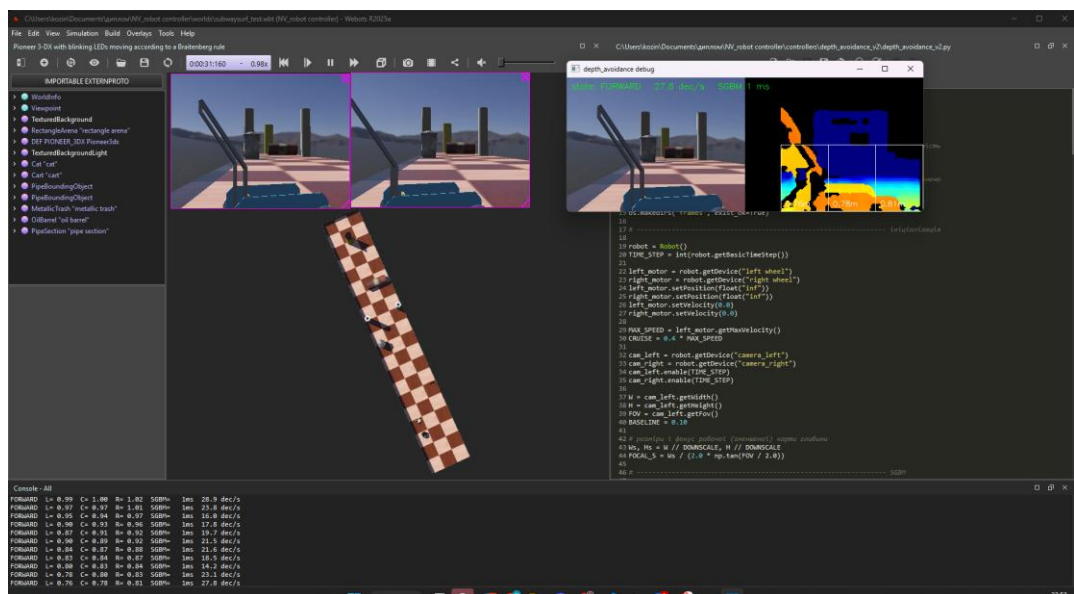


Рисунок 4.2 – Приклад демонстрації сцени в середовищі Webots

Під час кожного прогону карту глибини, поділ на зони, виміряні відстані d_L , d_C , d_R і поточний стан автомата виводили у вікно спостереження в реальному часі. Це давало змогу звіряти рішення робота з тим, що він фактично сприймає, і відрізняти правильну реакцію від випадкового збігу.

У відкритій частині арени з поодинокими перешкодами робот рухається вперед на крейсерській швидкості. Поки центральна зона вільна, пропорційне кермування лише плавно підвертає його до того боку, де простору більше, завдяки чому робот не наближається впритул до бічних перешкод, а тримає від них дистанцію. На підході до перешкоди попереду центральна відстань d_C спадає, і в разі перетину порога DANGER робот переходить у поворот, обертається до вільнішого боку, а після звільнення проходу повертається до прямолінійного руху. Цей перехід супроводжується зміною стану у вікні спостереження з FORWARD на TURN і назад на FORWARD.

У вузькому проході між двома об'єктами обидві бічні відстані малі й близькі за значенням, тому зміщення керування $bias$ виходить близьким до нуля, і пропорційне кермування утримує робота приблизно посередині проходу, не даючи притиснутись до жодного боку. Так підтверджується призначення пропорційного члена – у вузьких місцях він центрує робота.

Окремо перевірено підхід до стіни під гострим кутом, близько тридцяти градусів. У цій ситуації центральна зона спрямована вздовж стіни, тому її глибина велика і простір попереду видається вільним, хоча корпус наближається до стіни збоку; бічна відстань при цьому падає нижче порога SIDE_CRITICAL. Спрацьовує другий випадок переходу в поворот: робот розвертається від ближчого боку, не чекаючи, поки перешкода з'явиться в центральній зоні. Без цього правила робот доходив до стіни і тер об неї корпусом, що й стало підставою для його введення.

У глухому куті, де всі три зони показують малу відстань, робот переходить у задній хід, від'їжджає назад і повторює спробу повороту. У тіснішій ситуації, коли робот опиняється затиснутим між стіною та об'єктом, реактивний алгоритм без пам'яті схильний до зациклення, оскільки в однаковій обстановці щоразу обирає той самий напрям. Передбачений механізм виходу з пастки розриває цей цикл: після кількох невдалих циклів повороту і заднього ходу напрям повороту обирають випадково, і робот

зрештою знаходить вихід. Час виходу при цьому не є сталим і змінюється від прогону до прогону, що є прямим наслідком випадкового вибору напрямку.

Перевірено також реакцію на однотонні стіни без вираженої текстури. На таких поверхнях стереозіставлення не знаходить відповідностей, і відповідні пікселі карти глибини лишаються невалідними. Оскільки зона з малою часткою валідних пікселів у нижній частині трактується як заблокована, а не як вільний простір, робот зупиняє рух уперед і повертає, тобто не сприймає невидиму для стерео стіну як вільний напрям. На текстурованих перешкодах глибина визначається надійніше, тому робот реагує на них з більшої відстані й рухається плавніше. Ця відмінність підтверджує загальне обмеження стереометодів – залежність їхньої роботи від наявності текстури на поверхнях сцени.

Окремо оцінено швидкодію конвеєра, оскільки від неї залежить придатність методу для роботи в реальному часі. Карту глибини будують на зменшеному кадрі розміром 160 на 120 пікселів. За такого розміру одне обчислення карти методом SGBM на центральному процесорі триває близько 1 мс, що забезпечує частоту прийняття рішень приблизно 10 рішень за секунду. Цієї частоти достатньо для уникнення перешкод на крейсерській швидкості робота, що підтверджується його поведінкою в наведених сценаріях. Така швидкодія є додатковим обґрунтуванням вибору SGBM для контуру керування. Нейромережевий метод RAFT-Stereo, хоч і точніший, у реальному часі на центральному процесорі непридатний через значно вищу обчислювальну вартість, тому його застосовано лише для офлайн-порівняння точності на наборі даних, тоді як у задачі руху використано SGBM.

Проведене дослідження показує, що алгоритм уникнення коректно відпрацьовує всі розглянуті ситуації: рух у відкритому просторі, проходження вузьких місць, підхід до стіни під гострим кутом, вихід із глухих кутів і локальних пасток, а також реакцію на однотонні поверхні. Поведінка робота в кожному випадку відповідає закладеній логіці переходів між станами.

4.5 Аналіз результатів та обмеження системи

Отримані результати свідчать, що побудована система розв'язує поставлену задачу в межах змодельованого середовища: за стереопарою вдається будувати карту глибини, виділяти на ній зайняті й вільні зони і на цій основі вести робота серед перешкод без зіткнень. Кількісне оцінювання карт глибини та якісне спостереження за рухом доповнюють одне одного: перше показує, наскільки точно відновлюється геометрія сцени, друге – наскільки надійно ця геометрія перетворюється на правильні рішення керування. Разом вони підтверджують працездатність обраного підходу, але водночас окреслюють низку обмежень, що визначають межі його застосування.

Перше й найвідчутніше обмеження пов'язане із самою природою стереозору. Відновлення глибини спирається на пошук відповідних точок у лівому та правому зображеннях, а отже потребує вираженої текстури на поверхнях сцени. На однотонних стінах, гладких підлогах і за слабого освітлення відповідності не знаходяться, і відповідні ділянки карти глибини лишаються невалідними. У системі цей ефект частково знешкоджено: зона з малою часткою валідних пікселів трактується як заблокована, тому невидима для стерео стіна не сприймається як вільний простір. Проте таке рішення убезпечує від зіткнення ціною надмірної обережності – робот може зупинятися або повертати там, де перешкоди насправді немає, а просто бракує текстури.

Друге обмеження стосується точності відновленої глибини. Похибка стереовимірювання зростає з відстанню: для віддалених об'єктів навіть невелика неточність у зіставленні дає помітне розходження в оцінці глибини, тому надійно робот сприймає лише перешкоди в межах кількох метрів. До цього додається властива методу SGBM зернистість карти – на тонких структурах і межах об'єктів вона менш точна, ніж нейромережеві методи. Для

контуру керування такої точності достатньо, оскільки рішення приймаються за усередненими відстанями в зонах, а не за окремими пікселями, проте для задач, де потрібна детальна геометрія сцени, її було б замало.

Третє обмеження впливає з обраного представлення сцени трьома зонами. Такий поділ свідомо спрощує картину навколишнього простору заради швидкості й стійкості рішень, але разом з тим згладжує деталі. Дрібні, низькі або тонкі перешкоди, що потрапляють лише в невелику частину зони, можуть не знизити усереднену відстань настільки, щоб спрацював поріг, і залишитися непоміченими. Так само поза увагою лишаються об'єкти вище від поля зору камери, оскільки система оцінює простір тільки в межах кадру стереопари.

Четверте обмеження закладене в реактивному характері алгоритму уникнення. Робот приймає рішення лише за поточним кадром і не будує карти пройденого шляху, тому він не планує маршрут наперед і не гарантує досягнення цілі у складному середовищі. У просторі з розгалуженою структурою проходів такий алгоритм здатний завести робота в локальну пастку. Передбачений механізм випадкового вибору напрямку дозволяє з неї вибратися, однак він не оптимізує траєкторію, а лише розриває зациклення, через що час виходу непередбачуваний і за несприятливого збігу може виявитися тривалим.

П'яте обмеження стосується умов перевірки. Усі експерименти проведено у віртуальному середовищі, де стереопара ідеальна, освітлення кероване, а шуми сенсора й розмиття від руху відсутні. Реальна камера дає зашумлені кадри, її калібрування з часом порушується, а на відкритому повітрі додаються змінне освітлення та відблиски. Тому отримані результати слід розглядати як підтвердження правильності закладеної логіки, а не як оцінку поведінки системи на фізичному роботі: перенесення на апаратну платформу потребує окремого дослідження.

Висновки до розділу 4

У четвертому розділі проведено експериментальне дослідження розробленої системи та виконано аналіз отриманих результатів. Для кількісного оцінювання точності побудови глибини використано набір даних Middlebury 2021, який містить еталонні карти глибини та параметри калібрування для реальних сцен.

Виконано порівняння класичного методу SGBM і нейромережевого методу RAFT-Stereo за метриками AbsRel, RMSE, $\delta < 1,25$ та покриттям. Встановлено, що RAFT-Stereo забезпечує вищу точність відновлення глибини та повне покриття зображення, демонструючи стабільні результати навіть на складних сценах із малотекстурованими та відбивними поверхнями. Водночас метод SGBM характеризується нижчими обчислювальними витратами та придатністю до роботи в режимі реального часу.

Проведено дослідження роботи алгоритму уникнення перешкод у середовищі моделювання. Підтверджено коректність переходів між станами скінченного автомата та працездатність алгоритму в характерних ситуаціях, зокрема під час руху у відкритому просторі, проходження вузьких ділянок, обходу перешкод, виходу з глухих кутів і локальних пасток.

На основі проведених експериментів обґрунтовано використання методу SGBM у контурі керування мобільним роботом, оскільки його швидкодії достатньо для забезпечення своєчасного прийняття рішень. Нейромережевий метод RAFT-Stereo доцільно застосовувати в задачах, де пріоритетом є максимальна точність відновлення глибини.

Також визначено основні обмеження запропонованої системи, пов'язані із залежністю стереозору від текстури поверхонь, зростанням похибки на великих відстанях, спрощеним представленням сцени та реактивним характером алгоритму керування. Отримані результати підтверджують

працездатність запропонованого підходу та визначають напрями його подальшого розвитку.

ВИСНОВКИ

У дипломній роботі досліджено методи побудови карт глибини та уникнення перешкод для наземних дронів. Проведено аналіз особливостей автономної навігації наземних платформ, розглянуто основні типи сенсорів для отримання інформації про глибину сцени, а також виконано класифікацію класичних і нейромережових методів оцінки глибини та підходів до планування руху.

На основі проведеного аналізу обґрунтовано вибір стереозорового підходу та визначено структуру системи, яка поєднує модуль побудови карти глибини, модуль виявлення перешкод і підсистему прийняття рішень. Для формування карти глибини реалізовано два підходи: класичний алгоритм напівглобального зіставлення SGBM та нейромережовий метод RAFT-Stereo. Також розроблено алгоритм виявлення перешкод і реактивний алгоритм уникнення перешкод на основі скінченного автомата, призначений для роботи в умовах часткової невизначеності середовища.

Для перевірки працездатності запропонованого підходу створено модель мобільного робота та середовища в симуляторі Webots, а також проведено експериментальні дослідження з використанням набору даних Middlebury 2021. Виконано порівняння методів SGBM та RAFT-Stereo за показниками точності відновлення глибини. Встановлено, що нейромережовий метод RAFT-Stereo забезпечує вищу якість побудови карти глибини, тоді як метод SGBM характеризується меншими обчислювальними витратами та придатністю до використання в системах реального часу. Дослідження роботи алгоритму уникнення перешкод підтвердило його працездатність у типових сценаріях руху мобільного робота.

Практична цінність отриманих результатів полягає у можливості використання розроблених програмних компонентів під час створення систем

автономної навігації наземних роботизованих платформ з обмеженими обчислювальними ресурсами. Запропонований підхід може бути використаний як основа для подальшої інтеграції з повноцінними навігаційними стеками та системами локалізації.

Подальший розвиток роботи доцільно спрямувати на інтеграцію системи з ROS 2 та Nav2, використання глобального планування маршруту, розширення набору сенсорів за рахунок інерціальних датчиків або LiDAR, а також дослідження методів сенсорного синтезу та більш складних алгоритмів навігації у динамічному середовищі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hirschmuller H. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information // IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). 2005. P. 807–814.
2. Eigen D., Puhrsch C., Fergus R. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network // Advances in Neural Information Processing Systems (NeurIPS). 2014. Vol. 27. P. 2366–2374.
3. Ranftl R., Lasinger K., Hafner D., Schindler K., Koltun V. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer // IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). 2022. Vol. 44, № 3. P. 1623–1637.
4. Ranftl R., Bochkovskiy A., Koltun V. Vision Transformers for Dense Prediction // IEEE/CVF International Conference on Computer Vision (ICCV). 2021. P. 12179–12188.
5. Yang L., Kang B., Huang Z., Xu X., Feng J., Zhao H. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024. P. 10371–10381.
6. Yang L., Kang B., Huang Z., Zhao Z., Xu X., Feng J., Zhao H. Depth Anything V2 // Advances in Neural Information Processing Systems (NeurIPS). 2024. arXiv:2406.09414.
7. Godard C., Mac Aodha O., Firman M., Brostow G. J. Digging Into Self-Supervised Monocular Depth Estimation // IEEE/CVF International Conference on Computer Vision (ICCV). 2019. P. 3828–3838.
8. Mur-Artal R., Tardós J. D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras // IEEE Transactions on Robotics. 2017. Vol. 33, № 5. P. 1255–1262.

9. Labbé M., Michaud F. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation // *Journal of Field Robotics*. 2019. Vol. 36, № 2. P. 416–446.
10. Zhang Z. A Flexible New Technique for Camera Calibration // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000. Vol. 22, № 11. P. 1330–1334.
11. Hart P. E., Nilsson N. J., Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // *IEEE Transactions on Systems Science and Cybernetics*. 1968. Vol. 4, № 2. P. 100–107.
12. LaValle S. M. Rapidly-Exploring Random Trees: A New Tool for Path Planning : Technical Report TR 98-11. Iowa State University, 1998.
13. Karaman S., Frazzoli E. Sampling-Based Algorithms for Optimal Motion Planning // *International Journal of Robotics Research*. 2011. Vol. 30, № 7. P. 846–894.
14. Fox D., Burgard W., Thrun S. The Dynamic Window Approach to Collision Avoidance // *IEEE Robotics & Automation Magazine*. 1997. Vol. 4, № 1. P. 23–33.
15. Borenstein J., Koren Y. The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots // *IEEE Transactions on Robotics and Automation*. 1991. Vol. 7, № 3. P. 278–288.
16. Geiger A., Lenz P., Urtasun R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite // *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012. P. 3354–3361.
17. Silberman N., Hoiem D., Kohli P., Fergus R. Indoor Segmentation and Support Inference from RGBD Images // *European Conference on Computer Vision (ECCV)*. 2012. P. 746–760.
18. Macenski S., Foote T., Gerkey B., Lalancette C., Woodall W. Robot Operating System 2: Design, Architecture, and Uses in the Wild // *Science Robotics*. 2022. Vol. 7, № 66. eabm6074.

19. Macenski S., Martín F., White R., Ginés Clavero J. The Marathon 2: A Navigation System // IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020. P. 2718–2725.
20. Bradski G. The OpenCV Library // Dr. Dobb's Journal of Software Tools. 2000. Vol. 25, № 11. P. 120–125.
21. Scharstein D., Hirschmuller H., Kitajima Y., Krathwohl G., Nescic N., Wang X., Westling P. High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth // Pattern Recognition (GCPR). Lecture Notes in Computer Science. 2014. Vol. 8753. P. 31–42.
22. Michel O. Cyberbotics Ltd. Webots: Professional Mobile Robot Simulation // International Journal of Advanced Robotic Systems. 2004. Vol. 1, № 1. P. 39–42.
23. Paszke A., Gross S., Massa F. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library // Advances in Neural Information Processing Systems (NeurIPS). 2019. Vol. 32. P. 8024–8035.
24. Lipson L., Teed Z., Deng J. RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching // International Conference on 3D Vision (3DV). 2021. P. 218–227.
25. Mayer N., Ilg E., Hausser P., Fischer P., Cremers D., Dosovitskiy A., Brox T. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 4040–4048.
26. Khatib O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots // The International Journal of Robotics Research. 1986. Vol. 5, № 1. P. 90–98.

ДОДАТОК А ЛІСТИНГ КОДУ

```
depth_middlebury2021.py:
```

```
import cv2, numpy as np, os, re, sys, csv, glob
```

```
MAX_WIDTH = 1500
```

```
BAD_THRESH = 2.0
```

```
DEPTH_CAP_MARGIN = 1.05
```

```
def read_calib(path):
```

```
    d = {}
```

```
    with open(path) as fp:
```

```
        for line in fp:
```

```
            if "=" in line:
```

```
                k, v = line.strip().split("=", 1)
```

```
                d[k] = v
```

```
    nums = re.findall(r"[-+]?[0-9]*\.?[0-9]+(?:[eE][-+]?[0-9]+)?", d["cam0"])
```

```
    f = float(nums[0])
```

```
    doffs = float(d.get("doffs", 0))
```

```
    baseline = float(d.get("baseline", 1))
```

```
    ndisp = int(float(d.get("ndisp", 256)))
```

```
    return f, doffs, baseline, ndisp
```

```
def read_pfm(path):
```

```
    with open(path, "rb") as fp:
```

```
        color = fp.readline().decode().strip() == "PF"
```

```
        w, h = map(int, fp.readline().decode().split())
```

```

scale = float(fp.readline().decode())
data = np.fromfile(fp, ("<" if scale < 0 else ">") + "f")
data = np.reshape(data, (h, w, 3) if color else (h, w))
data = np.flipud(data).astype(np.float32).copy()
data[~np.isfinite(data)] = 0
return data

```

```

def build_matcher(ndisp, block=5):
    num_disp = max(16, int(np.ceil(ndisp / 16) * 16))
    left = cv2.StereoSGBM_create(
        minDisparity=0, numDisparities=num_disp, blockSize=block,
        P1=8 * block * block, P2=32 * block * block,
        disp12MaxDiff=1, uniquenessRatio=10,
        speckleWindowSize=100, speckleRange=32,
        mode=cv2.STEREO_SGBM_MODE_SGBM_3WAY)
    return left, num_disp

```

```

def compute_disparity(gL, gR, ndisp):
    left, num_disp = build_matcher(ndisp)
    disp = left.compute(gL, gR).astype(np.float32) / 16.0
    return disp, "SGBM"

```

```

def _empty_metrics(density):
    return dict(bad=np.nan, rmse_disp=np.nan, absrel=np.nan,
rmse_depth=np.nan,
                delta1=np.nan, delta2=np.nan, delta3=np.nan, density=float(density))

```

```

def evaluate(dis, gt, f, dofs, baseline, scale, bad_px):
    valid = dis > 0
    gv = gt > 0
    m = valid & gv
    res = _empty_metrics(valid.mean())
    if m.sum() == 0:
        return res, None

    dp, dg = dis[m], gt[m]
    err = np.abs(dp - dg)
    res["bad"] = float(np.mean(err > bad_px * scale))
    res["rmse_dis"] = float(np.sqrt(np.mean(err ** 2)))

    zp = baseline * f / (dp + dofs) / 1000.0
    zg = baseline * f / (dg + dofs) / 1000.0

    res["absrel"] = float(np.mean(np.abs(zp - zg) / zg))
    ratio = np.maximum(zp / zg, zg / zp)
    res["delta1"] = float(np.mean(ratio < 1.25))
    res["delta2"] = float(np.mean(ratio < 1.25 ** 2))
    res["delta3"] = float(np.mean(ratio < 1.25 ** 3))

    cap = float(zg.max()) * DEPTH_CAP_MARGIN
    zp_c, zg_c = np.clip(zp, 0, cap), np.clip(zg, 0, cap)
    res["rmse_depth"] = float(np.sqrt(np.mean((zp_c - zg_c) ** 2)))

    depth = np.zeros_like(dis)

```

```

depth[valid] = baseline * f / (disp[valid] + doffs) / 1000.0
return res, depth

```

```

def process_scene(scene):
    name = os.path.basename(os.path.normpath(scene))
    out = lambda fn: os.path.join(scene, fn)

    imgL = cv2.imread(out("im0.png"))
    imgR = cv2.imread(out("im1.png"))
    if imgL is None or imgR is None:
        return None

    f, doffs, baseline, ndisp = read_calib(out("calib.txt"))
    gt = read_pfm(out("disp0.pfm")) if os.path.exists(out("disp0.pfm")) else None

    s = 1.0
    if imgL.shape[1] > MAX_WIDTH:
        s = MAX_WIDTH / imgL.shape[1]
        imgL = cv2.resize(imgL, None, fx=s, fy=s,
interpolation=cv2.INTER_AREA)
        imgR = cv2.resize(imgR, None, fx=s, fy=s,
interpolation=cv2.INTER_AREA)
        f *= s; doffs *= s; ndisp = max(16, int(ndisp * s))
        if gt is not None:
            gt = cv2.resize(gt, (imgL.shape[1], imgL.shape[0]),
interpolation=cv2.INTER_NEAREST) * s

    gL = cv2.cvtColor(imgL, cv2.COLOR_BGR2GRAY)
    gR = cv2.cvtColor(imgR, cv2.COLOR_BGR2GRAY)

```

```

disp, algo = compute_disparity(gL, gR, ndisp)

res, depth = _empty_metrics((disp > 0).mean()), None
if gt is not None:
    res, depth = evaluate(disp, gt, f, doffs, baseline, s, BAD_THRESH)
if depth is None:
    valid = disp > 0
    depth = np.zeros_like(disp); depth[valid] = baseline * f / (disp[valid] +
doffs) / 1000.0

valid = disp > 0
dc = cv2.applyColorMap(cv2.normalize(disp, None, 0, 255,
cv2.NORM_MINMAX).astype(np.uint8),
cv2.COLORMAP_PLASMA)
cap = np.percentile(depth[valid], 95) if valid.any() else 1
zc = cv2.applyColorMap(cv2.normalize(np.clip(depth, 0, cap), None, 0, 255,
cv2.NORM_MINMAX).astype(np.uint8),
cv2.COLORMAP_VIRIDIS)
cv2.imwrite(out("disparity.png"), dc)
cv2.imwrite(out("depth_color.png"), zc)
cv2.imwrite(out("depth_result.png"), np.hstack([imgL, dc, zc]))
np.save(out("depth_pred.npy"), depth)

print(f"[{name}] {imgL.shape[1]}x{imgL.shape[0]} {algo} | "
f"bad {BAD_THRESH:g}={res['bad']*100:5.1f}%
RMSEd={res['rmse_disp']:5.2f}px "
f"AbsRel={res['absrel']:.3f} δ1={res['delta1']*100:4.0f}%
cov={res['density']*100:4.0f}%")
return dict(scene=name, **res)

```

```

def print_table(rows):
    hdr = ["scene", "bad2.0_%", "RMSE_disp_px", "AbsRel", "RMSE_depth_m",
           "δ<1.25_%", "coverage_%"]
    keys = ["bad", "rmse_disp", "absrel", "rmse_depth", "delta1", "density"]

    def fmt(r):
        return [
            r["scene"],
            f'{r["bad"]*100:.1f}' if np.isfinite(r["bad"]) else "-",
            f'{r["rmse_disp"]:.2f}' if np.isfinite(r["rmse_disp"]) else "-",
            f'{r["absrel"]:.3f}' if np.isfinite(r["absrel"]) else "-",
            f'{r["rmse_depth"]:.3f}' if np.isfinite(r["rmse_depth"]) else "-",
            f'{r["delta1"]*100:.1f}' if np.isfinite(r["delta1"]) else "-",
            f'{r["density"]*100:.0f}' if np.isfinite(r["density"]) else "-",
        ]

    cells = [fmt(r) for r in rows]

    def agg(key, fn):
        vals = [r[key] for r in rows if np.isfinite(r[key])]
        return fn(vals) if vals else np.nan

    cells.append(["—" * 8] + ["—" * 6] * (len(hdr) - 1))
    for label, fn in [("Average", np.mean), ("Median", np.median)]:
        a = {k: agg(k, fn) for k in keys}
        cells.append([label] + fmt(dict(scene=label, **a))[1:])

```

```
widths = [max(len(hdr[i]), max(len(row[i]) for row in cells)) for i in
range(len(hdr))]
```

```
line = lambda c: " ".join(str(c[i]).ljust(widths[i]) for i in range(len(hdr)))
```

```
print("\n" + line(hdr))
```

```
print(" ".join("-" * w for w in widths))
```

```
for row in cells:
```

```
    print(line(row))
```

```
def main():
```

```
    root = sys.argv[1] if len(sys.argv) > 1 else "."
```

```
    scenes = sorted(os.path.dirname(p) for p in glob.glob(os.path.join(root, "**",
"calib.txt"), recursive=True))
```

```
        if os.path.exists(os.path.join(os.path.dirname(p), "im0.png")))
```

```
print(f"Found: {len(scenes)}\n")
```

```
rows = [r for s in scenes if (r := process_scene(s))]
```

```
print_table(rows)
```

```
def agg(key, fn):
```

```
    vals = [r[key] for r in rows if np.isfinite(r[key])]
```

```
    return fn(vals) if vals else float("nan")
```

```
with open("metrics.csv", "w", newline="", encoding="utf-8-sig") as fp:
```

```
    w = csv.writer(fp)
```

```
    w.writerow(["scene", "bad2.0_%", "RMSE_disp_px", "AbsRel",
"RMSE_depth_m",
" $\delta < 1.25$ _%", " $\delta < 1.25^2$ _%", " $\delta < 1.25^3$ _%", "coverage_%"])
```

```

for r in rows:
    w.writerow([r["scene"],
                 f'{r["bad"]*100:.2f}', f'{r["rmse_disp"]:.3f}',
                 f'{r["absrel"]:.4f}', f'{r["rmse_depth"]:.4f}',
                 f'{r["delta1"]*100:.2f}', f'{r["delta2"]*100:.2f}',
                 f'{r["delta3"]*100:.2f}',
                 f'{r["density"]*100:.1f}'])
for label, fn in [("Average", np.mean), ("Median", np.median)]:
    w.writerow([label,
                 f'{agg("bad", fn)*100:.2f}', f'{agg("rmse_disp", fn):.3f}',
                 f'{agg("absrel", fn):.4f}', f'{agg("rmse_depth", fn):.4f}',
                 f'{agg("delta1", fn)*100:.2f}', f'{agg("delta2", fn)*100:.2f}',
                 f'{agg("delta3", fn)*100:.2f}', f'{agg("density", fn)*100:.1f}'])
print(f"\nOutput table -> metrics.csv")

```

```

if __name__ == "__main__":
    main()
stereo_raft.py:
## 1. Встановлення conda
"""

!pip install -q condacolab
from shutil import which
import condacolab
if which("conda") is None:
    condacolab.install()

"""## 2. Збирання середовища"""

```

```

# Commented out IPython magic to ensure Python compatibility.
!git clone --recursive https://github.com/princeton-vl/RAFT-Stereo.git
# %cd /content/RAFT-Stereo
!conda env create -f environment.yaml
print("\nСередовище зібрано. Далі: Runtime -> Run after")

"""## 3. Завантаження ваг моделі"""

# Commented out IPython magic to ensure Python compatibility.
# %cd /content/RAFT-Stereo
!pip install imageio
!bash download_models.sh

"""## 4. Завантаження і підготовка датасету"""
# %cd /content/RAFT-Stereo
import os, re, urllib.request, numpy as np, cv2

BASE = "https://vision.middlebury.edu/stereo/data/scenes2021/data"
SCENES =
["artroom1", "artroom2", "bandsaw1", "bandsaw2", "chess1", "chess2", "chess3",
"curule1", "curule2", "curule3", "ladder1", "ladder2", "octogons1", "octogons2",
"pendulum1", "pendulum2", "podium1", "skates1", "skates2", "skiboats1", "skiboats2",
,
"skiboats3", "traproom1", "traproom2"]
TARGET_W = 1500 # та сама ширина, що в локальному
batch_depth.py

```

```
RAW, PREP = "mb2021_raw", "mb2021"
```

```
opener = urllib.request.build_opener(); opener.addheaders = [("User-
Agent","Mozilla/5.0")]
urllib.request.install_opener(opener)
```

```
def read_pfm(p):
    with open(p, "rb") as f:
        color = f.readline().decode().strip() == "PF"
        w, h = map(int, f.readline().decode().split())
        sc = float(f.readline().decode())
        d = np.fromfile(f, ("<" if sc < 0 else ">") + "f").reshape((h, w, 3) if color
else (h, w))
        return np.flipud(d).copy()
```

```
os.makedirs(PREP, exist_ok=True)
for i, sc in enumerate(SCENES, 1):
    rd, pd = os.path.join(RAW, sc), os.path.join(PREP, sc)
    os.makedirs(rd, exist_ok=True); os.makedirs(pd, exist_ok=True)
    for fn in ["im0.png", "im1.png", "calib.txt", "disp0.pfm"]:
        dst = os.path.join(rd, fn)
        if not (os.path.exists(dst) and os.path.getsize(dst) > 0):
            try: urllib.request.urlretrieve(f"{BASE}/{sc}/{fn}", dst)
            except Exception as e: print(" !", sc, fn, e)
    cal = {}
    for line in open(os.path.join(rd, "calib.txt")):
        if "=" in line: k, v = line.strip().split("=", 1); cal[k] = v
    f = float(re.findall(r"[-+]?\\d*\\.?\\d+(?:[eE][-+]?\\d+)?", cal["cam0"])[0])
    doffs = float(cal.get("doffs", 0)); baseline = float(cal.get("baseline", 1))
```

```

imgL      =      cv2.imread(os.path.join(rd,      "im0.png"));      imgR      =
cv2.imread(os.path.join(rd, "im1.png"))
gt = read_pfm(os.path.join(rd, "disp0.pfm")); gt[~np.isfinite(gt)] = 0
s = min(1.0, TARGET_W / imgL.shape[1])
if s < 1.0:
    imgL      =      cv2.resize(imgL,      None,      fx=s,      fy=s,
interpolation=cv2.INTER_AREA)
    imgR      =      cv2.resize(imgR,      None,      fx=s,      fy=s,
interpolation=cv2.INTER_AREA)
    gt      =      cv2.resize(gt,      (imgL.shape[1],      imgL.shape[0]),
interpolation=cv2.INTER_NEAREST) * s
    f *= s; doffs *= s
    cv2.imwrite(os.path.join(pd, "im0.png"), imgL)
    cv2.imwrite(os.path.join(pd, "im1.png"), imgR)
    np.save(os.path.join(pd, "gt_disp.npy"), gt.astype(np.float32))
    np.save(os.path.join(pd,      "calib.npy"),      np.array([f,      doffs,      baseline],
dtype=np.float64))
    print(f"[{i}/{len(SCENES)}]      {sc}:      {imgL.shape[1]}x{imgL.shape[0]}
f={f:.1f} doffs={doffs:.1f} B={baseline:.1f}")

```

"""## 5. Запуск RAFT-Stereo по всіх сценах"""

```

!source activate raftstereo && python demo.py \
--restore_ckpt models/raftstereo-sceneflow.pth \
--corr_implementation alt --mixed_precision \
-l="mb2021/*/im0.png" -r="mb2021/*/im1.png" \
--output_directory raft_out --save_numpy

```

"""## 6. Метрики проти еталона"""

```

import numpy as np, os, glob, csv, cv2

def metrics_for(scene_dir, disp_npy):
    f, doffs, baseline = np.load(os.path.join(scene_dir, "calib.npy"))
    gt = np.load(os.path.join(scene_dir, "gt_disp.npy"))
    disp = np.abs(np.load(disp_npy).squeeze().astype(np.float32))    # RAFT:
    диспаратність від'ємна
    if disp.shape != gt.shape:
        disp = cv2.resize(disp, (gt.shape[1], gt.shape[0]),
        interpolation=cv2.INTER_NEAREST)
    valid = disp > 0
    depth = np.zeros_like(disp); depth[valid] = (baseline * f) / (disp[valid] + doffs)
    / 1000.0
    gv = gt > 0; gtd = np.zeros_like(gt); gtd[gv] = (baseline * f) / (gt[gv] + doffs)
    / 1000.0
    m = valid & gv; pr, tr = depth[m], gtd[m]
    return (float(np.mean(np.abs(pr - tr) / tr)), float(np.sqrt(np.mean((pr - tr) **
    2))),
    float(np.mean(np.maximum(pr / tr, tr / pr) < 1.25)), float(valid.mean()))

rows = []
print(f'{'scene':<12} {'AbsRel':>8} {'RMSE,m':>9} {'d<1.25':>9} {'cov.':>8}")
for sd in sorted(glob.glob("mb2021/*")):
    name = os.path.basename(sd); npy = f'raft_out/{name}.npy'
    if not os.path.exists(np):
        print(f'пропуск {name}: немає {np}'); continue
    a, r, d, c = metrics_for(sd, npy); rows.append((name, a, r, d, c))
    print(f'{'name':<12} {'a':>8.3f} {'r':>9.3f} {'100*d:>8.1f}%{'100*c:>7.0f}%")

```

```

if rows:
    A=np.mean([x[1] for x in rows]); R=np.mean([x[2] for x in rows])
    D=np.mean([x[3] for x in rows]); C=np.mean([x[4] for x in rows])
    print("-"*46)

print(f'{'СЕРЕДНЕ':<12} {A:>8.3f} {R:>9.3f} {100*D:>8.1f}% {100*C:>7.0f}%')
    with open("raft_metrics.csv", "w", newline="") as fp:
        w = csv.writer(fp);
w.writerow(["scene","AbsRel","RMSE_m","delta<1.25_%" ,"coverage_%"])
        for n,a,r,d,c in rows: w.writerow([n, f'{a:.4f}', f'{r:.4f}', f'{100*d:.1f}',
f'{100*c:.0f}'])
    print("\nТаблиця -> raft_metrics.csv")
    try:
        from google.colab import files; files.download("raft_metrics.csv")
    except Exception:
        pass
    depth_avoidance_v2:
import os
import time
import numpy as np
import cv2
from controller import Robot

# ----- показ і продуктивність

SHOW_WINDOW = True
VIZ_EVERY = 3
SAVE_EVERY = 0
DEPTH_VIS_MAX = 3.0

```

DOWNSCALE = 2

os.makedirs("frames", exist_ok=True)

----- ініціалізація

robot = Robot()

TIME_STEP = int(robot.getBasicTimeStep())

left_motor = robot.getDevice("left wheel")

right_motor = robot.getDevice("right wheel")

left_motor.setPosition(float("inf"))

right_motor.setPosition(float("inf"))

left_motor.setVelocity(0.0)

right_motor.setVelocity(0.0)

MAX_SPEED = left_motor.getMaxVelocity()

CRUISE = 0.4 * MAX_SPEED

cam_left = robot.getDevice("camera_left")

cam_right = robot.getDevice("camera_right")

cam_left.enable(TIME_STEP)

cam_right.enable(TIME_STEP)

W = cam_left.getWidth()

H = cam_left.getHeight()

FOV = cam_left.getFov()

BASELINE = 0.10

```

# розміри і фокус робочої (зменшеної) карти глибини
Ws, Hs = W // DOWNSCALE, H // DOWNSCALE
FOCAL_S = Ws / (2.0 * np.tan(FOV / 2.0))

# ----- SGBM

NUM_DISP = 64 // DOWNSCALE
NUM_DISP = max(16, (NUM_DISP // 16) * 16) # кратність 16
BLOCK = 5

sgbm = cv2.StereoSGBM_create(
    minDisparity=0,
    numDisparities=NUM_DISP,
    blockSize=BLOCK,
    P1=8 * BLOCK * BLOCK,
    P2=32 * BLOCK * BLOCK,
    uniquenessRatio=10,
    speckleWindowSize=60,
    speckleRange=2,
    disp12MaxDiff=1,
    mode=cv2.STEREO_SGBM_MODE_SGBM_3WAY,
)

Z_MIN = FOCAL_S * BASELINE / NUM_DISP

# ----- параметри руху

DANGER = 0.7      # м: центр ближче -> TURN
SAFE = 1.4        # м: центр далі (і боки вільні) -> вихід з TURN

```

```

SIDE_CRITICAL = 0.4 # м: бік ближче -> TURN від нього
CRITICAL = 0.3      # м: усі зони ближче -> BACKUP
PROCESS_EVERY = 6   # кроків симуляції на одне рішення (6*16мс ~ 10.4
Гц)
BACKUP_TICKS = 12   # тривалість заднього ходу, у рішеннях
TURN_TIMEOUT = 50   # рішень у TURN без виходу -> примусовий
BACKUP
STEER_GAIN = 0.6    # сила пропорційного кермування (0 = вимкнено)
STEER_RANGE = 1.5   # м: відстані понад цю вважаються "повністю
вільно"
MIN_VALID_FRAC = 0.25 # частка валідних пікселів у ближній частині
зони
SAFE_RELAX_TICKS = 15 # після стількох рішень у TURN досить центру
> DANGER
ESCAPE_AFTER = 2     # циклів TURN->BACKUP поспіль до режиму
втечі
stuck_cycles = 0

print(f"setup: cam {W}x{H}, SGBM on {Ws}x{Hs}, numDisp={NUM_DISP},
"
      f"Z_min={Z_MIN:.2f} m")
print(f"params:          DANGER={DANGER}          SAFE={SAFE}
SIDE_CRITICAL={SIDE_CRITICAL} "
      f"CRITICAL={CRITICAL} PROCESS_EVERY={PROCESS_EVERY} "
      f"BACKUP_TICKS={BACKUP_TICKS}
TURN_TIMEOUT={TURN_TIMEOUT} "
      f"STEER_GAIN={STEER_GAIN}
STEER_RANGE={STEER_RANGE}")

```

----- обробка кадрів

```
def grab_bgr(camera):
    buf = camera.getImage()
    img = np.frombuffer(buf, np.uint8).reshape((H, W, 4))
    return cv2.cvtColor(img, cv2.COLOR_BGRA2BGR)

def compute_depth(frame_l, frame_r):
    gl = cv2.cvtColor(frame_l, cv2.COLOR_BGR2GRAY)
    gr = cv2.cvtColor(frame_r, cv2.COLOR_BGR2GRAY)
    if DOWNSCALE > 1:
        gl = cv2.resize(gl, (Ws, Hs), interpolation=cv2.INTER_AREA)
        gr = cv2.resize(gr, (Ws, Hs), interpolation=cv2.INTER_AREA)
    disp = sgbm.compute(gl, gr).astype(np.float32) / 16.0
    depth = np.full(disp.shape, np.inf, dtype=np.float32)
    valid = disp > 0.5
    depth[valid] = FOCAL_S * BASELINE / disp[valid]
    return depth

def zone_distances(depth):
    roi = depth[Hs // 2:, NUM_DISP:]
    zones = np.array_split(roi, 3, axis=1)
    dists = []
    for z in zones:
        near = z[int(z.shape[0] * 0.6):, :]
        if np.isfinite(near).mean() < MIN_VALID_FRAC:
            dists.append(0.0)
```

```

        continue
    finite_mask = np.isfinite(z)
    dists.append(float(np.percentile(z[finite_mask], 10)))
return dists

# ----- візуалізація

def depth_to_color(depth):
    finite = np.isfinite(depth)
    norm = np.zeros_like(depth)
    norm[finite] = np.clip(depth[finite], 0, DEPTH_VIS_MAX) /
    DEPTH_VIS_MAX
    gray = (255 * (1.0 - norm)).astype(np.uint8)
    color = cv2.applyColorMap(gray, cv2.COLORMAP_JET)
    color[~finite] = (0, 0, 0)
    return color

def make_panel(frame_l, depth, dists, state, fps, calc_ms):
    d_l, d_c, d_r = dists
    vis = depth_to_color(depth)
    if DOWNSCALE > 1:
        vis = cv2.resize(vis, (W, H), interpolation=cv2.INTER_NEAREST)

    x0, y0 = NUM_DISP * DOWNSCALE, H // 2
    zone_w = (W - x0) // 3
    cv2.rectangle(vis, (x0, y0), (W - 1, H - 1), (255, 255, 255), 1)
    for i in (1, 2):

```

```

        cv2.line(vis, (x0 + i * zone_w, y0), (x0 + i * zone_w, H - 1),
                  (255, 255, 255), 1)
    for i, d in enumerate((d_l, d_c, d_r)):
        label = f"{d:.2f}m" if d > 0 else "BLOCK"
        cv2.putText(vis, label, (x0 + i * zone_w + 8, H - 10),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 255, 255), 1)

    panel = np.hstack([frame_l, vis])
    cv2.putText(panel, f'state: {state}  {fps:.1f} dec/s  SGBM {calc_ms:.0f} ms',
                 (8, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 1)
    return panel

def save_snapshot(tag, frame_l, frame_r, panel):
    cv2.imwrite(f'frames/{tag}_left.png', frame_l)
    cv2.imwrite(f'frames/{tag}_right.png', frame_r)
    cv2.imwrite(f'frames/{tag}_panel.png', panel)
    print(f'saved frames/{tag}_*.png')

# ----- СТАН АВТОМАТА

state = "FORWARD"
turn_dir = 1      # +1 = ВЛІВО, -1 = ВПРАВО
backup_left = 0
turn_ticks = 0
step_count = 0
decision_count = 0
last_decision_wall = time.perf_counter()

```

```
fps = 0.0
```

```
# ----- ГОЛОВНИЙ ЦИКЛ
```

```
while robot.step(TIME_STEP) != -1:
```

```
    step_count += 1
```

```
    if step_count % PROCESS_EVERY != 0:
```

```
        continue
```

```
    decision_count += 1
```

```
    now = time.perf_counter()
```

```
    dt = now - last_decision_wall
```

```
    fps = (1.0 / dt) if dt > 0 else 0.0
```

```
    last_decision_wall = now
```

```
    frame_l = grab_bgr(cam_left)
```

```
    frame_r = grab_bgr(cam_right)
```

```
    t0 = time.perf_counter()
```

```
    depth = compute_depth(frame_l, frame_r)
```

```
    calc_ms = (time.perf_counter() - t0) * 1000.0
```

```
    d_left, d_center, d_right = zone_distances(depth)
```

```
# ---- переходи між станами -----
```

```
if state == "FORWARD":
```

```
    if d_center < CRITICAL and d_left < CRITICAL and d_right < CRITICAL:
```

```
        state = "BACKUP"
```

```
        backup_left = BACKUP_TICKS
```

```

elif d_center < DANGER:
    turn_dir = 1 if d_left > d_right else -1
    state = "TURN"
    turn_ticks = 0
elif d_left < SIDE_CRITICAL or d_right < SIDE_CRITICAL:
    turn_dir = -1 if d_left < d_right else 1
    state = "TURN"
    turn_ticks = 0
elif state == "TURN":
    turn_ticks += 1
    center_ok = (d_center > SAFE
                 or (turn_ticks > SAFE_RELAX_TICKS and d_center >
DANGER))
    if (center_ok
        and d_left > SIDE_CRITICAL and d_right > SIDE_CRITICAL):
        state = "FORWARD"
        stuck_cycles = 0
    elif turn_ticks > TURN_TIMEOUT:
        stuck_cycles += 1
        state = "BACKUP"
        backup_left = BACKUP_TICKS * min(stuck_cycles, 3)
    elif d_center < CRITICAL and d_left < CRITICAL and d_right <
CRITICAL:
        state = "BACKUP"
        backup_left = BACKUP_TICKS

elif state == "BACKUP":
    backup_left -= 1
    if backup_left <= 0:

```

```

    if stuck_cycles >= ESCAPE_AFTER:
        turn_dir = 1 if np.random.rand() < 0.5 else -1
    else:
        turn_dir = 1 if d_left > d_right else -1
    state = "TURN"
    turn_ticks = 0

# ---- ШВИДКОСТІ КОЛІС -----
if state == "FORWARD":
    dl = min(d_left, STEER_RANGE)
    dr = min(d_right, STEER_RANGE)
    bias = (dr - dl) / STEER_RANGE
    v_l = float(np.clip(CRUISE * (1.0 + STEER_GAIN * bias), 0.0,
MAX_SPEED))
    v_r = float(np.clip(CRUISE * (1.0 - STEER_GAIN * bias), 0.0,
MAX_SPEED))
elif state == "TURN":
    v_l = -turn_dir * 0.3 * CRUISE
    v_r = turn_dir * 0.3 * CRUISE
else:
    v_l = v_r = -0.3 * CRUISE

left_motor.setVelocity(v_l)
right_motor.setVelocity(v_r)

# ---- візуалізація і збереження -----
need_viz = SHOW_WINDOW and decision_count % VIZ_EVERY == 0
need_save = SAVE_EVERY and decision_count % SAVE_EVERY == 0
if need_viz or need_save:

```

```

panel = make_panel(frame_l, depth,
                    (d_left, d_center, d_right), state, fps, calc_ms)
tag = f'{int(robot.getTime() * 1000):07d}'
if need_save:
    save_snapshot(tag, frame_l, frame_r, panel)
if need_viz:
    cv2.imshow("depth_avoidance debug", panel)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("s"):
        save_snapshot(tag, frame_l, frame_r, panel)

print(f'{state:7s} L={d_left:5.2f} C={d_center:5.2f} R={d_right:5.2f} '
      f'SGBM={calc_ms:4.0f}ms {fps:4.1f} dec/s")

```