МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова праця
на правах рукопису

Деркаченко Анатолій Олегович

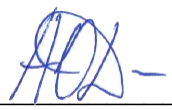
УДК 681.5.015:004.94:621.865.8

Дисертація

МЕТОДИ ІДЕНТИФІКАЦІЇ ПАРАМЕТРІВ РУХУ ЕЛЕМЕНТІВ КАРКАСНИХ КОНСТРУКЦІЙ

151 Автоматизація та комп’ютерно-інтегровані технології
15 Автоматизація та приладобудування

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання результатів і текстів інших авторів мають посилання на відповідне джерело

 А.О. Деркаченко

Науковий керівник Поливода Оксана Валеріївна – кандидат технічний наук, доцент

Хмельницький 2026

АНОТАЦІЯ

Деркаченко А.О. Методи ідентифікації параметрів руху елементів каркасних конструкцій. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології. – Херсонський національний технічний університет, Хмельницький, 2026.

Дисертацію присвячено розробленю методів ідентифікації параметрів руху елементів каркасних конструкцій на основі сферичних паралельних механізмів (СПМ) та їх програмно-апаратній реалізації в системах керування.

У першому розділі проведено аналіз механізмів паралельної структури (МПС), їх типів, класифікації та принципів побудови. Розглянуто традиційні класифікаційні підходи за структурою кінематичних ланцюгів, методом перетворення рухів, типом кінематичних пар, кінематичною топологією та розташуванням приводів. Визначено, що основними галузями застосування МПС є аерокосмічна техніка, транспортні засоби, медичне обладнання, робототехнічні та вимірювальні системи, спортивні тренажери, сонячні трекери. Проаналізовано методи математичного моделювання маніпуляторів, зокрема метод замкнутого векторного контуру та метод перетворення координат, а також технічні засоби реалізації систем керування, серед яких сенсорні системи, обчислювальні платформи реального часу, виконавчі приводи та інтерфейси зв'язку. Розглянуто методи ідентифікації параметрів руху МПС до яких відносяться оптичне, інерціальне, магнітне відстеження, візуально-інерціальне злиття даних та модельно-орієнтоване відстеження. Обґрунтовано доцільність застосування гібридних методів, що поєднують сенсорні дані з математичними моделями та необхідність розроблення спеціалізованого програмного забезпечення для СПМ.

У другому розділі розроблено математичну модель сферичного паралельного механізму, що охоплює геометричний опис прототипу, модель орієнтації платформи на основі матричних перетворень, інкрементальний підхід до визначення кутового положення, врахування геометричних та кінематичних

обмежень. Запропоновано багатокроковий алгоритм фільтрації для вибору єдиної фізично допустимої конфігурації механізму з-поміж множини можливих розв'язків систем нелінійних рівнянь. Розроблено методику визначення робочої області СПМ на основі послідовного аналізу просторових положень платформи та розв'язання задачі оберненої кінематики. Проведено розрахунковий експеримент для типових траєкторій руху (спіраль, коло, зиґзаґ, квадрат), який підтвердив адекватність математичної моделі та її придатність для інтеграції в автоматизовані системи керування.

У третьому розділі запропоновано метод ідентифікації параметрів руху елементів СПМ на основі алгоритму комплементарного фільтра, адаптованого для систем керування з обмеженими обчислювальними ресурсами. Алгоритм поєднує дані гіроскопа та акселерометра для оцінки кута нахилу рухомої платформи, зменшуючи вплив дрейфу гіроскопа та шумів акселерометра. Експериментальна перевірка на дослідному зразку СПМ підтвердила працездатність методу, а динамічна похибка не перевищувала допустимого значення. Розроблено програмно-апаратну платформу керування СПМ у режимі реального часу, що забезпечує інтелектуальну обробку відеоданих для відстеження положення об'єкта керування (руки або обличчя) за допомогою камери. Система реалізована у вигляді дворівневої архітектури: клієнтський графічний інтерфейс виконує захоплення та первинну обробку відеопотоку з використанням бібліотек OpenCV та MediaPipe, а серверна частина відповідає за обчислення оберненої кінематики та формування керуючих сигналів для серводвигунів. Розроблено апаратну частину на основі мікроконтролера Raspberry Pi Pico RP2040 з інерціальним модулем MPU6050, OLED-дисплеєм та клавіатурою для автономної роботи.

У четвертому розділі досліджено практичні аспекти застосування СПМ у складі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів. Проаналізовано методи сканування (лазерне, фотограмметрія, ультразвуковий контроль) та методи планування траєкторій на основі геометричних моделей і алгоритмів пошуку шляху (A^*, RRT). Розглянуто стратегії сканування (спіральна, кругова, зигзагоподібна, квадратна траєкторії)

відповідно до форми автоклава. Розроблено алгоритм обробки хмари точок, отриманої методом фотограмметрії, що включає п'ять етапів фільтрації на основі аналізу координат нормалей (*PCA*, *k*-найближчих сусідів) та координат точок для виділення потенційних дефектів поверхні. Запропоновано структуру автоматизованої системи діагностики з використанням СПМ, що включає операторський інтерфейс, центральний контролер, модулі планування траєкторій та керування рухом, систему збору та обробки даних.

Ключові слова: сферичний паралельний механізм, рухома платформа, ідентифікація, параметри руху, методи фільтрації даних, комплементарний фільтр, математичне моделювання, кінематика, комп'ютерний зір, датчики, аналіз зображень, діагностика, автоматизована система керування, механізм паралельної структури, планування і відстеження траєкторій, алгоритм, мікроконтролер, автоматизація, контролер реального часу, комп'ютерно-інтегрована система, серводвигуни, автономні системи, модуль, автоклав, каркасні конструкції.

Список публікацій:

1. Поливода О.В., Деркаченко А.О., Поливода В.В. Автоматизоване розпізнавання дефектів поверхні автоклавів з використанням аналізу хмар точок. *Прикладні питання математичного моделювання*. 2025. Том 8, Ч 1. С. 147 – 155. https://doi.org/10.32782/mathematical-modelling/2025-8-1-14

2. Деркаченко А.О., Поливода О.В. Використання комплементарного фільтра для ідентифікації параметрів руху елементів сферичного паралельного механізма // *Проблеми інформатизації та управління*. 2025. № 1 (81). С. 23–29. https://doi.org/10.18372/2073-4751.81.20125
https://jrnl.nau.edu.ua/index.php/PIU/issue/view/1004

3. Деркаченко А. О., Поливода О. В. Автоматизована система діагностики внутрішніх поверхонь промислових автоклавів з використанням сферичного паралельного механізму. *Вісник Херсонського національного технічного університету*. 2025. №2. Ч.1. (93). С.48-55. https://doi.org/10.35546/kntu2078-4481.2025.2.1.6

4. Деркаченко А.О., Поливода О.В. Розробка алгоритму визначення положення платформи сферичного паралельного механізму в 3D-просторі та його програмна реалізація. *Вісник Херсонського національного технічного університету*. 2025. Том 1 №3 (94). С.79–87. DOI: https://doi.org/10.35546/kntu2078-4481.2025.3.1.9

5. Деркаченко Т.О., Поливода О.В. Деркаченко А.О. Розробка комплексної математичної моделі системи просторового позиціонування сонячного трекера як механізму паралельної структури. *Прикладні питання математичного моделювання*. 2025. Том 8, (Ч.2.). С.93–101. DOI: https://doi.org/10.32782/mathematical-modelling/2025-8-2-10

Апробації результатів дисертації:

1. Y. Lebedenko, O. Polyvoda, A. Derkachenko, Y. Modlo, S. Demishonkova and Y. Pylypenko, "Research of Control Systems for Robotic Spatial Planning Platforms," *2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES)*, Kremenchuk, Ukraine, 2022, 1-4. (Scopus) DOI: https://doi.org/10.1109/MEES58014.2022.10005765

2. A. Derkachenko, O. Polyvoda, Y. Lebedenko and K. Kalinina, "Research of Control Methods of a Spherical Parallel Mechanism Using Intelligent Data Processing," *2023 IEEE 5th International Conference on Modern Electrical and Energy System (MEES)*, Kremenchuk, Ukraine, 2023, 1-5. (Scopus) https://doi.org/10.1109/MEES61502.2023.10402530

3. Деркаченко А.О., Поливода О.В. Аналіз засобів вимірювання параметрів руху елементів каркасних конструкцій. *Матеріали X Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління присвяченої Дню ракетно-космічної галузі України*. Херсон – Хмельницький, 2023. С. 15-18.

4. Деркаченко А.О., Поливода О.В., Русанов С.А., Лебеденко Ю.О. Керування приводами сферичного паралельного механізму з використанням мови програмування MAXSCRIPT. *Матеріали V всеукраїнської науково-технічної конференції «Інформаційні технології та програмування».* (Херсон, 27 грудня 2023 р.). Хмельницький: ХНТУ. С. 17-21.

5. Деркаченко А.О., Поливода О.В. Математична модель руху елементів каркасних конструкцій // *Матеріали XI Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління присвяченої Дню ракетно-космічної галузі України: Збірник наукових праць.* Херсон-Хмельницький: Видавництво ФОП Вишемирський В.С., 2024. С. 41-45.

6. Дмитрієв Д.О., Русанов С.А., Деркаченко А.О., Лебеденко Ю.О., Поливода О.В. Свідоцтво про реєстрацію авторського права на твір №111283 «Комп'ютерна програма «ToolsSM»» від 24.01.2022 р. https://sis.nipo.gov.ua/uk/search/detail/1694611/

7. Деркаченко А.О., Поливода О.В. Свідоцтво про реєстрацію авторського права на твір № 141922 Комп'ютерна програма «СПМ API Server» (Сервер централізованого керування модулями обчислення параметрів руху сферичного паралельного механізму (СПМ))» («СПМ API Server») https://sis.nipo.gov.ua/uk/search/detail/1901901/

ABSTRACT

Derkachenko A.O. Methods for Identification of Motion Parameters of Elements of Framework Structures. – Qualifying Scientific Work Submitted as a Manuscript.

Dissertation submitted for the degree of Doctor of Philosophy in Specialty 151 – Automation and Computer-Integrated Technologies. – Kherson National Technical University, Khmelnytskyi, 2026.

The dissertation is dedicated to the development of methods for identifying motion parameters of elements of framework structures based on spherical parallel mechanisms (SPMs) and their software and hardware implementation in control systems.

The first chapter presents an analysis of parallel structure mechanisms, their types, classifications, and design principles. Traditional classification approaches based on the structure of kinematic chains, motion transformation methods, types of kinematic pairs, kinematic topology, and actuator placement are considered. It is determined that the main application areas of parallel structure mechanisms include aerospace engineering, transportation systems, medical equipment, robotic and measurement systems, sports simulators, and solar trackers. Methods of mathematical modeling of manipulators are analyzed, including the closed-loop vector method and coordinate transformation method, as well as hardware components of control systems such as sensor systems, real-time computing platforms, actuators, and communication interfaces. Methods for identifying motion parameters of parallel structure mechanisms are analyzed, including optical, inertial, and magnetic tracking, visual-inertial data fusion, and model-based tracking. The expediency of applying hybrid methods combining sensor data with mathematical models and the necessity of developing specialized software for spherical parallel mechanisms are substantiated.

The second chapter develops a mathematical model of a spherical parallel mechanism, including a geometric description of the prototype, a platform orientation model based on matrix transformations, an incremental approach to determining angular position, and consideration of geometric and kinematic constraints. A multi-stage filtering algorithm is proposed for selecting a unique physically feasible

configuration of the mechanism from multiple possible solutions of a system of nonlinear equations. A methodology for determining the workspace of the spherical parallel mechanism based on sequential analysis of platform spatial positions and solving the inverse kinematics problem is developed. A computational experiment involving typical motion trajectories (spiral, circular, zigzag, and square) confirms the adequacy of the mathematical model and its suitability for integration into automated control systems.

The third chapter proposes a method for identifying motion parameters of spherical parallel mechanism elements based on a complementary filter algorithm adapted for control systems with limited computational resources. The algorithm combines gyroscope and accelerometer data to estimate the inclination angle of the moving platform while reducing the influence of gyroscope drift and accelerometer noise. Experimental validation performed on an SPM prototype confirmed the operability of the proposed method, while the dynamic error did not exceed the permissible value. A real-time software and hardware control platform for spherical parallel mechanisms was developed, providing intelligent video data processing for tracking the position of a controlled object (hand or face) using a camera. The system is implemented as a two-level architecture in which the client graphical interface performs video capture and preliminary processing using OpenCV and MediaPipe libraries, while the server side is responsible for inverse kinematics calculations and generation of control signals for servo drives. The hardware subsystem is based on the Raspberry Pi Pico RP2040 microcontroller and includes an MPU6050 inertial measurement unit, an OLED display, and a keypad for autonomous operation.

The fourth chapter investigates practical aspects of applying spherical parallel mechanisms within an automated system for diagnostics of internal surfaces of industrial autoclaves. Scanning methods, including laser scanning, photogrammetry, and ultrasonic inspection, as well as trajectory planning methods based on geometric models and path-planning algorithms (A^*, RRT), are analyzed. Scanning strategies such as spiral, circular, zigzag, and square trajectories are considered according to the autoclave geometry. An algorithm for processing point clouds obtained through photogrammetry is developed. The algorithm includes five filtering stages based on the

analysis of normal vector coordinates (*PCA*, *k-nearest neighbors*) and point coordinates for detecting potential surface defects. The structure of an automated diagnostic system using a spherical parallel mechanism is proposed, including an operator interface, a central controller, trajectory planning and motion control modules, and a data acquisition and processing subsystem.

Keywords: spherical parallel mechanism, moving platform, identification, motion parameters, data filtering methods, complementary filter, mathematical modeling, kinematics, computer vision, sensors, image analysis, diagnostics, automated control system, parallel mechanism, trajectory planning and tracking, algorithm, microcontroller, automation, real-time controller, computer-integrated system, servo motors, autonomous systems, module, autoclave, framework structures.

List of Publications:

1. Polyvoda O.V., Derkachenko A.O., Polyvoda V.V. Automated recognition of autoclave surface defects using point cloud analysis. *Applied Issues of Mathematical Modeling*. 2025. Vol. 8, Part 1. 147–155. DOI: https://doi.org/10.32782/mathematical-modelling/2025-8-1-14

2. Derkachenko A.O., Polyvoda O.V. Use of complementary filter for identification of motion parameters of elements of spherical parallel mechanism. *Problems of Informatization and Management*. 2025. 1 (81). 23–29. DOI: https://doi.org/10.18372/2073-4751.81.20125
https://jrnl.nau.edu.ua/index.php/PIU/issue/view/1004

3. Derkachenko A.O., Polyvoda O.V. Automated system for diagnostic internal surfaces of industrial autoclaves using a spherical parallel mechanism. *Bulletin of Kherson National Technical University*. 2025. 2. Part 1. (93). 48–55. DOI: https://doi.org/10.35546/kntu2078-4481.2025.2.1.6

4. Derkachenko A.O., Polyvoda O.V. Development of an algorithm for determining the position of a spherical parallel mechanism platform in 3D space and its software implementation. *Bulletin of Kherson National Technical University*. 2025. Vol. 1, 3 (94). 79–87. DOI: https://doi.org/10.35546/kntu2078-4481.2025.3.1.9

5. Derkachenko T.O., Polyvoda O.V., Derkachenko A.O. Development of a complex mathematical model of the solar tracker spatial positioning system as a parallel structure mechanism. *Applied Issues of Mathematical Modeling*. 2025. Vol. 8, Part 2. 93–101. DOI: https://doi.org/10.32782/mathematical-modelling/2025-8-2-10

Approbations of the Dissertation Results:

1. Y. Lebedenko, O. Polyvoda, A. Derkachenko, Y. Modlo, S. Demishonkova and Y. Pylypenko, Research of Control Systems for Robotic Spatial Planning Platforms. *2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES)*, Kremenchuk, Ukraine, 2022, 1-4. (Scopus) DOI: https://doi.org/10.1109/MEES58014.2022.10005765

2. A. Derkachenko, O. Polyvoda, Y. Lebedenko and K. Kalinina, Research of Control Methods of a Spherical Parallel Mechanism Using Intelligent Data Processing. *2023 IEEE 5th International Conference on Modern Electrical and Energy System (MEES).* Kremenchuk, Ukraine, 2023, 1-5. (Scopus) DOI: https://doi.org/10.1109/MEES61502.2023.10402530

3. Derkachenko A.O., Polyvoda O.V. Analysis of measuring tools for motion parameters of frame structure elements. *Proceedings of the X All-Ukrainian Scientific and Practical Conference of Higher Education Students and Young Scientists on Automatic Control Dedicated to the Day of the Rocket and Space Industry of Ukraine*. Kherson – Khmelnytskyi, 2023. 15-18.

4. Derkachenko A.O., Polyvoda O.V., Rusanov S.A., Lebedenko Yu.O. Control of spherical parallel mechanism drives using the MAXSCRIPT programming language. *Proceedings of the V All-Ukrainian Scientific and Technical Conference "Information Technologies and Programming"*. (Kherson, December 27, 2023). Khmelnytskyi: KhNTU. 17-21.

5. Derkachenko A.O., Polyvoda O.V. Mathematical model of motion of frame structure elements. *Proceedings of the XI All-Ukrainian Scientific and Practical Conference of Higher Education Students and Young Scientists on Automatic Control Dedicated to the Day of the Rocket and Space Industry of Ukraine: Collection of Scientific Works.* Kherson-Khmelnytskyi: FOP Vyshemyrskyi V.S. Publishing House, 2024. 41-45.

6. Dmitriev D.O., Rusanov S.A., Derkachenko A.O., Lebedenko Yu.O., Polyvoda O.V. Copyright registration certificate for work 111283 "Computer Program 'ToolsSM'" dated 24.01.2022. https://sis.nipo.gov.ua/uk/search/detail/1694611/

7. Derkachenko A.O., Polyvoda O.V. Copyright registration certificate for work 141922 Computer Program "SPM API Server" (Centralized control server for motion parameter calculation modules of a spherical parallel mechanism (SPM)) ("SPM API Server"). https://sis.nipo.gov.ua/uk/search/detail/1901901/

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ 14
ВСТУП 15
РОЗДІЛ 1 АНАЛІЗ МЕХАНІЗМІВ ПАРАЛЕЛЬНОЇ СТРУКТУРИ ТА СФЕР ЇХ ЗАСТОСУВАННЯ 21
1.1. Механізми паралельної структури: типи, класифікація, принципи побудови та сфери застосування 21
1.1.1. Характеристики схем маніпуляторів промислових роботів 24
1.1.2. Класифікація механізмів паралельної структури 25
1.1.3. Галузі використання механізмів паралельної структури 28
1.2. Методи математичного моделювання маніпуляторів 31
1.2.1. Метод замкнутого векторного контуру 32
1.2.2. Метод перетворення координат 33
1.3. Технічні засоби реалізації систем керування 35
1.3.1. Сенсорні системи вимірювання положення та орієнтації 37
1.3.2. Обчислювальні платформи реального часу 42
1.3.3. Виконавчі приводи та інтерфейси зв'язку 43
1.4. Методи ідентифікації параметрів руху МПС 43
1.4.1. Оптичне відстеження 44
1.4.2. Інерціальне відстеження 44
1.4.3. Магнітне відстеження 45
1.4.4. Візуально-інерціальне злиття даних 45
1.4.5. Модельно-орієнтоване відстеження 46
1.5. Висновки до розділу 1 46
РОЗДІЛ 2 МАТЕМАТИЧНА МОДЕЛЬ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ 50
2.1. Опис прототипу сферичного паралельного механізму 50
2.2. Формування геометричної моделі механізму 53
2.3. Урахування геометричних обмежень та допустимих конфігурацій 55
2.4. Модель орієнтації платформи через матричні перетворення 61
2.5. Визначення кутового положення на основі інкрементального підходу . 65

2.6. Визначення робочої області сферичного паралельного механізму 66
2.7. Розрахунковий експеримент для сферичного паралельного механізму. 69
2.8. Висновки до розділу 2 73
РОЗДІЛ 3 АЛГОРИТМІЧНА ТА ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ .. 75
3.1. Алгоритм комплементарного фільтра для ідентифікації параметрів руху 75
3.2. Загальна структура програмного забезпечення 82
3.3. Програмна реалізація алгоритму визначення положення платформи ... 85
3.4. Апаратна частина програмно-апаратної платформи 93
3.5. Висновки до розділу 3 95
РОЗДІЛ 4 ВИКОРИСТАННЯ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ В СИСТЕМАХ АВТОМАТИЧГОЇ ДІАГНОСТИКИ ПРОМИСЛОВИХ ПОВЕРХОНЬ 97
4.1. Методи сканування поверхонь автоклавів 98
4.2. Методи планування траєкторій сканування на основі геометричних моделей 99
4.3. Вибір стратегії сканування внутрішній поверхні автоклава 101
4.4. Отримання та побудова хмари точок за даними фотограмметричного сканування 105
4.5. Алгоритм обробки хмари точок за даними фотограмметричного сканування 107
4.6. Структура автоматизованої системи сканування внутрішньої поверхні автоклава 116
4.7. Висновки до розділу 4 118
ВИСНОВКИ 121
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 123
ДОДАТОК А 134
ДОДАТОК Б 136
ДОДАТОК В 137

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ВІС – візуально-інерціальні системи;

ВО – виконавчий орган;

ІВМ – інерціальний вимірювальний модуль;

КП – кінематична пара;

МПС – механізм паралельної структури;

СПМ – сферичний паралельний механізм;

ФНЧ – фільтр низької частоти;

ШІМ – широтно-імпульсна модуляція.

ВСТУП

Сучасний розвиток робототехнічних систем та мехатронних комплексів характеризується зростанням вимог до точності позиціонування, швидкодії та адаптивності виконавчих механізмів. Механізми паралельної структури (МПС) посідають особливе місце серед таких систем завдяки підвищеній жорсткості, точності та здатності сприймати значні навантаження [1-5]. Окремим класом МПС є сферичні паралельні механізми (СПМ), які забезпечують реалізацію просторових обертальних рухів навколо фіксованого центра з трьома ступенями свободи [6-12].

Однією з ключових проблем під час експлуатації СПМ є забезпечення високої точності ідентифікації параметрів руху елементів механізму в режимі реального часу. Сигнали з інерційних датчиків містять шуми та систематичні похибки, що потребує застосування ефективних методів фільтрації та злиття даних [13,14]. Існуючі підходи на основі фільтрів Калмана, Маджвіка та Махоні забезпечують високу точність, однак характеризуються значною обчислювальною складністю, що обмежує їх застосування в системах реального часу з обмеженими апаратними ресурсами [15-18]. Тому розробка методу ідентифікації параметрів руху елементів СПМ адаптованого для систем керування з обмеженими обчислювальними ресурсами, є актуальним науково-технічним завданням.

Важливим аспектом практичного впровадження методів ідентифікації параметрів руху є їх програмна реалізація у складі спеціалізованих систем керування СПМ. Ефективне функціонування таких систем потребує інтеграції математичних моделей механізму, алгоритмів керування, засобів збору й обробки сенсорних даних, а також інтерфейсів взаємодії з користувачем [19]. Сучасні програмні засоби комп’ютерного зору та розподілені клієнт-серверні архітектури дають змогу реалізувати керування СПМ у режимі реального часу, забезпечуючи відстеження об’єктів, формування керуючих впливів та візуалізацію параметрів роботи механізму [20-22]. У зв’язку з цим актуальним є розроблення програмно-апаратної платформи, яка поєднує методи ідентифікації руху, математичні моделі

СПМ та засоби інтелектуальної обробки даних у єдиній автоматизованій системі керування.

Практичне застосування СПМ у складі автоматизованих систем діагностики внутрішніх поверхонь промислового обладнання, відкриває нові можливості для безконтактного контролю технічного стану. Розробка алгоритмів планування траєкторій сканування, методів обробки вимірювальних даних для виявлення дефектів поверхонь та створення програмно-апаратної платформи керування СПМ є важливими складовими вирішення цієї проблеми.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження проводилось в рамках тематичного плану науково-дослідних робіт відповідно теми «Оптимізація управління розподіленими рухомими об’єктами», кафедри автоматизації, робототехніки і мехатроніки, Херсонського національного технічного університету.

Мета і задачі дослідження. Метою роботи є підвищення ефективності процесу ідентифікації параметрів руху елементів каркасних конструкцій на основі сферичних паралельних механізмів шляхом розробки комплексу математичних моделей, алгоритмів та програмно-апаратних засобів ідентифікації на основі інтеграції геометричного моделювання, методів фільтрації вимірювальної інформації та комп'ютерного зору, що забезпечує високу точність позиціонування виконавчих органів у реальному часі та надійність мехатронних систем в умовах обмежених обчислювальних ресурсів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести аналіз існуючих механізмів паралельної структури, методів математичного моделювання, технічних засобів реалізації систем керування та методів ідентифікації параметрів руху МПС.

2. Розробити математичну модель сферичного паралельного механізму, що забезпечує визначення положення та орієнтації рухомої платформи, обчислення відносних кутів переміщення ланок і розв’язання задачі формування керуючих впливів для виконавчих приводів в автоматизованих робототехнічних системах.

3. Розробити метод ідентифікації параметрів руху елементів СПМ на основі даних інерційних сенсорів, адаптований для систем керування реального часу з обмеженими обчислювальними ресурсами.

4. Розробити програмно-апаратну платформу керування СПМ, що забезпечує інтелектуальну обробку відеоданих для відстеження положення об'єкта керування, визначення його просторових координат, передачу даних між програмними компонентами системи, обчислення параметрів руху механізму та формування керуючих сигналів для виконавчих приводів у режимі реального часу.

5. Дослідити практичні аспекти застосування сферичного паралельного механізму у складі автоматизованої системи діагностики внутрішніх промислових поверхонь та перевірити працездатність методів обробки даних для виявлення дефектів шляхом комп’ютерного моделювання.

Об'єкт дослідження. Системи і процеси керування сферичними паралельними механізмами.

Предмет дослідження. Математичні моделі, методи ідентифікації параметрів руху в процесі керування елементами каркасних конструкцій на основі сферичних паралельних механізмів.

Методи дослідження. В основу досліджень покладено методи математичного моделювання просторових механізмів, матричного аналізу для опису орієнтації твердого тіла, методи цифрової обробки сигналів та комплементарної фільтрації, методи комп'ютерного зору, методи обробки хмар точок, а також методи комп'ютерного моделювання та експериментальних досліджень.

Наукова новизна одержаних результатів. Наукова новизна дисертаційної роботи визначається сукупністю результатів, спрямованих на підвищення точності ідентифікації параметрів руху елементів СПМ, а саме:

– вперше запропоновано математичну модель сферичного паралельного механізму, в якій розв'язання оберненої задачі кінематики зводиться до визначення координат точок з'єднання рухомої платформи з виконавчими приводами як результату перетину сферичної поверхні та кола в

площині основи механізму із застосуванням апарату аналітичної геометрії, що забезпечує безпосереднє обчислення параметрів руху механізму для формування керуючих впливів та дозволяє відмовитися від громіздких матричних перетворень систем координат, що дозволяє зменшити обчислювальну складність порівняно з традиційними підходами.

– удосконалено метод ідентифікації параметрів руху елементів СПМ на основі комплементарного фільтра, адаптованого для автоматизованих систем керування з обмеженими обчислювальними ресурсами;

– вперше запропоновано застосування сферичного паралельного механізму як засобу позиціонування діагностичного інструменту в автоматизованій системі контролю внутрішніх поверхонь промислових автоклавів, що дозволило підвищити маневреність та точність сканування в обмеженому просторі.

– отримали подальший розвиток методи обробки хмар точок для діагностики поверхонь промислового обладнання, які за рахунок багатоетапної фільтрації на основі аналізу координат нормалей та просторових координат точок дозволяють підвищити точність виділення потенційних дефектів;

– удосконалено структуру автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів з використанням СПМ, яка передбачає інтеграцію модулів планування траєкторій сканування, керування рухом на основі зворотної кінематики та інтелектуальної обробки відеоданих.

Практичне значення одержаних результатів. Використання результатів дисертаційного дослідження дозволяє: підвищити точність керування сферичними паралельними механізмами в режимі реального часу за рахунок застосування комплементарного фільтра; реалізувати безконтактне дистанційне керування СПМ на основі інтелектуальної обробки відеоданих; автоматизувати процес діагностики внутрішніх поверхонь промислових автоклавів з використанням хмар точок та фотограмметрії.

Результати дисертаційного дослідження впроваджені в виробничі процеси ПП "Наш продукт" (Акт від 05.11.2025 р.) для автоматизації діагностики внутрішніх поверхонь промислових автоклавів із використанням СПМ як засобу

позиціонування діагностичного інструменту для виявлення потенційних дефектів поверхонь.

Особистий внесок здобувача. Всі положення, що виносяться на захист, належать особисто здобувачу і не містять результатів, ідей або розробок, що належать співавторам, разом з якими опубліковані наукові праці. У спільних публікаціях здобувачу належать: [23] – реалізовано математичний апарат обчислення кутів повороту платформи на основі відстежуваних рухів голови та перетворення їх у керуючі сигнали для сервоприводів; [24] – розроблено структуру автоматизованої системи діагностики з використанням СПМ, що включає операторський інтерфейс, центральний контролер, модулі планування траєкторій та керування рухом. Виконано моделювання руху СПМ за типовими траєкторіями; [25] – обґрунтовано вибір акселерометрів та гіроскопів як оптимальних за співвідношенням точність/ціна для задач ідентифікації параметрів руху елементів СПМ; [26] – виконано розрахунковий експеримент кінематичних властивостей сферичного механізму для типових траєкторій (коло, спіраль) та визначено закони керування приводних ланок; [27] – автором особисто проаналізовано існуючі методи математичного моделювання механізмів паралельної структури (метод замкнутого векторного контуру, метод перетворення координат); [28] – запропоновано розглядати сонячні трекери як механізми з паралельною структурою; [29] – запропоновано багатокроковий алгоритм фільтрації для вибору єдиної фізично допустимої конфігурації механізму, що включає перевірку належності точок колу, дотримання мінімальних відстаней між ланками, відсутність перетину та вибір найближчої до попереднього стану комбінації; [30] – запропоновано використовувати комплементарний фільтр в системі управління СПМ для обробки даних з акселерометра та гіроскопа; [31] – за пропоновано метод обробки хмар точок для діагностики поверхонь промислового обладнання, на основі багатоетапної фільтрації на основі аналізу координат нормалей; [32] – зовнішній вигляд інтерфейсу, [33] – програмно реалізовано сервер централізованого керування модулями обчислення параметрів руху сферичного паралельного механізму; [34] – запропоновано математичну модель СПМ для визначення положення та

орієнтації рухомої платформи на основі розв'язання зворотної задачі кінематики з використанням законів аналітичної геометрії.

Апробація результатів дисертації. Матеріали дисертації оприлюднено на міжнародних та всеукраїнських конференціях, зокрема: 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), м. Кременчук, 2022 р.; 2023 IEEE 5th International Conference on Modern Electrical and Energy System (MEES), м. Кременчук, 2023 р.; X Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих вчених з автоматичного управління, м. Херсон – Хмельницький, 2023 р.; V Всеукраїнській науково-технічній конференції «Інформаційні технології та програмування», м. Херсон, 2023 р.; XI Всеукраїнській науково-практичній конференції здобувачів вищої освіти та молодих вчених з автоматичного управління, м. Херсон – Хмельницький, 2024 р.

Публікації. Основні наукові результати дисертаційного дослідження опубліковано в наукових публікаціях, з яких: 5 статтей у наукових фахових виданнях України, 2 публікації у наукових виданнях, що цитуються наукометричною базою Scopus, а також 3 публікації в матеріалах міжнародних та всеукраїнських конференцій, 2 авторських свідоцтва на комп’ютерні програми.

Структура та обсяг дисертації. Загальний обсяг дисертації 196 сторінок, у тому числі 133 сторінок основного тексту. Робота складається зі вступу, 4 розділів, висновків та 3 додатки; містить 43 рисунка та 2 таблиці, список використаних джерел з 110 найменування.

РОЗДІЛ 1 АНАЛІЗ МЕХАНІЗМІВ ПАРАЛЕЛЬНОЇ СТРУКТУРИ ТА СФЕР ЇХ ЗАСТОСУВАННЯ

Серед широкого класу інженерних систем важливе місце займають каркасні конструкції (або каркасні/структурні механічні системи), що складаються зі стержневих елементів, об'єднаних у єдину несучу структуру для забезпечення необхідної жорсткості та передачі навантажень [35-37].

Розвиток методів структурного синтезу механічних систем привів до формування окремого класу механізмів паралельної структури.

На цьому рівні узагальнення знаходиться ширший клас механічних систем, серед яких виділяються механізми паралельної структури. Вони є підмножиною робототехнічних і механічних систем, у яких рухома платформа з’єднана з основою не єдиним послідовним кінематичним ланцюгом, а декількома незалежними паралельними кінематичними гілками, що утворюють замкнені контури [38-41].

Окремим випадком цього класу є сферичні паралельні механізми, які становлять спеціальний випадок механізмів паралельної структури. Їх ключовою особливістю є те, що всі обертальні рухи відбуваються навколо спільного центра або еквівалентної точки, а кінематика системи описується переважно сферичними рухами з трьома ступенями свободи, що відповідають орієнтації тіла в просторі [42].

1.1. Механізми паралельної структури: типи, класифікація, принципи побудови та сфери застосування

На відміну від класичних серійних маніпуляторів, у яких приводні ланки поєднані послідовно та похибки кожної ланки накопичуються вздовж структури, МПС мають кілька незалежних кінематичних ланцюгів, що одночасно підтримують рухому платформу. Прикладом МПС є платформа Стюарта, кінематична схема якої наведена на рис. 1.1.

Через таку архітектуру сумарна похибка істотно менша завдяки взаємній компенсації деформацій та геометричних відхилень у гілках [38, 39].

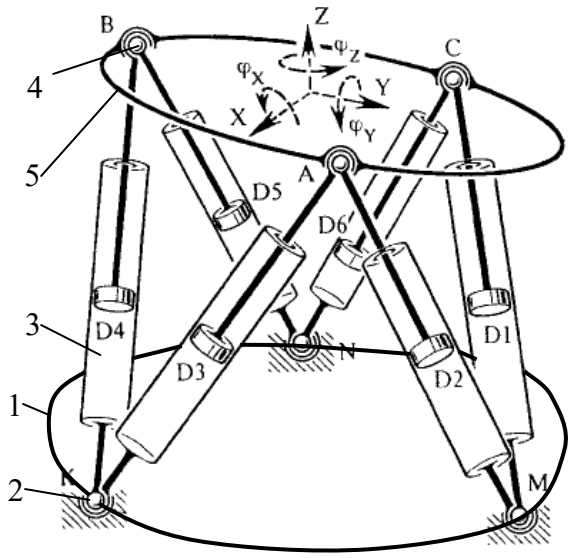


Рис. 1.1 – Приклад МПС

Така конструкція забезпечує підвищену жорсткість, точність позиціонування, стійкість до зовнішніх збурень, а також кращу динаміку завдяки розподілу навантаження між декількома приводами, що підтверджено сучасними аналітичними та оглядовими дослідженнями 2021–2025 років [1, 2, 4, 5]. Завдяки таким особливостям вони займають важливе місце серед сучасних робототехнічних і мехатронних систем.

Першим відомим прикладом сучасного паралельного механізму стала робота Д. Стюарта (1965), який запропонував шестивісну платформу для моделювання рухів літальних апаратів. Подальший розвиток, як відзначають дослідники [43, 44], стосувався структурного синтезу, виявлення сінгулярностей та методів їх уникнення. Розвиток високошвидкісних МПС відбувся завдяки роботам Р. Клавеля, який створив високошвидкісний Delta-маніпулятор який став основою для сучасних систем високошвидкісного монтажу електроніки (pick-and-place), описаних у роботах [3, 4] та ґрунтовним дослідженням [44], що систематизували підходи до аналізу та синтезу паралельних механізмів.

Особливості побудови кінематичних структур, аналізу та поділ механізмів за структурними ознаками механізмів паралельної структури висвітлено у працях з мехатроніки та теорії механізмів і машин у сучасних українських джерелах, зокрема в роботі [38], де значна увага приділяється класифікації маніпуляторів, принципам керування та особливостям побудови замкнених структур. У навчальних матеріалах [39] розглянуто засадничі принципи побудови механізмів, ланцюгів та структурних груп , а джерело [40] поглиблює ці положення аналізом структурних елементів механізмів, принципів формування їхніх кінематичних схем і критеріями, що застосовуються для класифікації складних багатоланкових систем. Додатково важливий внесок також у розвиток теорії й прикладного застосування МПС становить монографія [42].

На міжнародному рівні широке узагальнення методів моделювання та застосування МПС подано в оглядових роботах [1, 2, 4, 5]. Зокрема, в роботі [1] підкреслено, що у сучасних паралельних роботах активно впроваджуються інтелектуальні методи, включно з машинним навчанням, яке дозволяє виконувати компенсацію похибок, оптимізацію траєкторій та підвищення точності кінематичного моделювання у реальному часі, особливо для систем зі складними нелінійними деформаціями та багатовимірними просторами станів [1]. Такі методи стають ключовими для високоточних МПС, де класичні аналітичні моделі не забезпечують потрібної точності або швидкодії.

Сучасні огляди також показують чітку тенденцію розвитку від класичних шестивісних гексаподів до переналаштовуваних структур, гібридних паралельно-послідовних механізмів та високошвидкісних платформ із покращеною жорсткістю, динамікою та адаптивністю [2, 4, 5]. Зокрема, в [5] наголошується, що ре-конфігуровані МПС дозволяють змінювати структуру кінематичних ланцюгів для оптимізації робочого простору, уникнення сингулярностей або підвищення навантажувальної здатності, що робить їх перспективними для робототехніки нового покоління.

До класичних представників МПС належать платформа Стюарта, маніпулятор типу «Дельта», гексаподи, триподні структури та їх численні варіанти.

1.1.1. Характеристики схем маніпуляторів промислових роботів

У промисловій робототехніці паралельні механізми розглядаються як високоточні структури з короткими кінематичними ланцюгами, що обмежує накопичення похибок [2, 38-40]. Схема маніпулятора промислового робота визначається кількістю ступенів свободи, типом кінематичних ланцюгів та способом зв’язування рухомої платформи з базою.

Основні характеристики МПС включають:

1. Жорсткість конструкції. Завдяки замкненим контурам жорсткість набагато вища, ніж у серійних маніпуляторів, що підвищує точність під час навантаження. Що підтверджується численними дослідженнями, де показано, що жорсткість МПС може перевищувати серійні аналоги у 2–6 разів, залежно від конфігурації та напрямку навантаження. У сучасній літературі 2020–2025 рр. наводяться розширені методи підвищення жорсткості, включно із застосуванням машинного навчання для компенсації нелінійностей [1, 3, 45-47].

2. Підвищена точність позиціонування. Помилки приводів частково компенсуються геометрією механізму: рух платформи є результатом «усереднення» руху кількох ланцюгів. одна з ключових характеристик МПС, що детально аналізується у дисертаційних та оглядових роботах, а також у моделях гідравлічних та паралельно-послідовних маніпуляторів [4, 48, 49]. Сучасні дослідження демонструють, що структурний синтез і запобігання сінгулярностям забезпечують підвищення точності й повторюваності МПС [43, 44].

3. Кращі динамічні властивості. Мала маса рухомої платформи та рівномірний розподіл приводних зусиль дають змогу розвивати великі прискорення. особливо для Delta-маніпуляторів, детально розглянуті в роботах [3 та 50], а також у сучасних структурах з гібридними кінематичними ланками, що поєднують переваги паралельних і послідовних механізмів [3, 46, 47].

4. Схемна універсальність. Платформа Стюарта реалізує 6-ступеневу систему, тоді як Delta — високошвидкісні 3-DOF, а гібридні структури поєднують серійні та паралельні сегменти.

5. Компактність. Паралельні конструкції дозволяють мінімізувати габарити, що важливо для медичних роботів, автономних платформ і компактних вимірювальних систем.

Таким чином, сукупність цих властивостей робить МПС особливо зручними для задач, де критичною є точність і стабільність руху, а також швидкість., що узгоджується з результатами систематичних оглядів 2021–2025 років [1, 2, 4, 5].

1.1.2. Класифікація механізмів паралельної структури

Класифікація механізмів паралельної структури є ключовим етапом їх систематизації, аналізу та подальшого застосування у технологічному обладнанні. Необхідність такої класифікації зумовлена значною різноманітністю кінематичних схем, конструктивних рішень і функціональних можливостей МПС, які відрізняються принципами перетворення рухів, просторовими конфігураціями та характеристиками виконавчих органів [38-40, 42].

Традиційні класифікаційні підходи, що застосовуються для технологічного обладнання (зокрема, за типом технологічного процесу, рівнем автоматизації, розташуванням інструмента або організації руху), не можуть бути безпосередньо перенесені на МПС через їхню специфічну просторову кінематику та здатність формувати рух за рахунок взаємодії кількох паралельних кінематичних гілок.

Це зумовлює необхідність створення окремої класифікаційної системи, побудованої з урахуванням загальних класифікаційних ознак та особливостей побудови МПС. Зважаючи на те, що типовими елементами таких конструкцій є: базова платформа, рухома платформа, 3–6 кінематичних ланцюгів, що містять обертальні та поступальні кінематичні пари, а також шарніри різних типів (U-, S-, R-, P-пари), формується необхідний ступінь свободи системи.

Виходячи з цього класифікаційна система МПС може бути представлена у вигляді чотирьох групами характеристик:

1. Функціональні ознаки — типи рухів, технологічне призначення.
2. Кінематичні ознаки — структура ланцюгів, ступені свободи, види кінематичних пар.

3. Конструктивні ознаки — виконання штанг, компоновка приводів, типи шарнірів.

4. Експлуатаційні ознаки — точність, динамічні характеристики, жорсткість, навантажувальна здатність.

Ця класифікація дозволяє коректно систематизувати різні типи МПС і задати основу для подальшого проєктування на основі ознак, притаманних саме механізмам паралельної структури [42].

1.1.2.1. Класифікація за структурою кінематичних ланцюгів

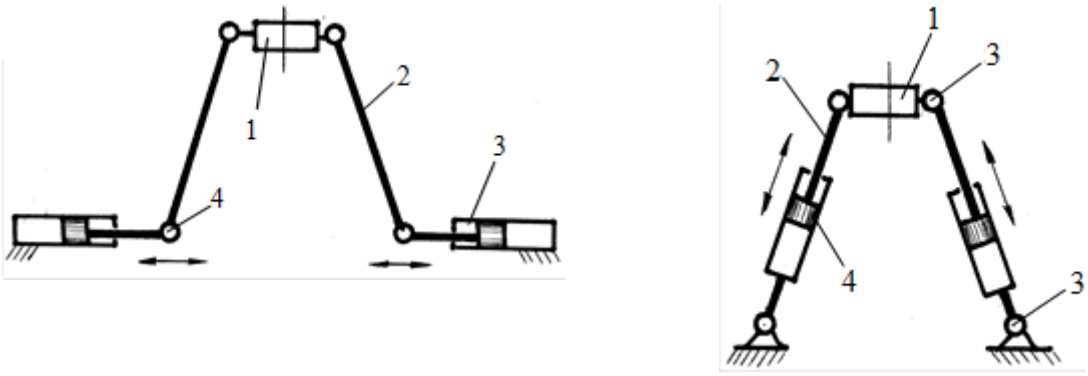
Однією з базових класифікаційних ознак є поділ МПС за кількістю кінематичних ланцюгів і співвідношенням між кількістю ланцюгів та числом ступенів вільності виконавчого органу (ВО). Існує дві основні групи МПС [42]: повнопаралельні механізми у яких кількість ступенів свободи ВО дорівнює кількості кінематичних ланцюгів із незалежним приводом та неповнопаралельні механізми у яких ВО має більше ступенів вільності, ніж кількість кінематичних ланцюгів; кожен ланцюг може містити кілька приводів.

Такий поділ відображає взаємозв’язок між геометрією механізму і його руховими можливостями, що є критично важливим для подальшої конструктивної оптимізації [40, 42].

1.1.2.2. Класифікація за методом перетворення рухів

Залежно від способу створення рухів ВО, тобто за методом перетворення рухів, МПС класифікують як: механізми зі зміною координат опорних шарнірів, у яких штанги мають постійну довжину (рис. 1.2 а), де положення ВО визначається зміщенням опорних точок шарнірів та механізми зі зміною довжин кінематичних ланок, де штанги мають змінну довжину завдяки використанню гідроциліндрів або лінійних електроприводів (рис. 1.2 б). На рис. 1.2 прийнято наступні позначення: 1 – ВО, 2 – кінематичні ланки – штанги, 3- вісь штоку гідроциліндру, 4 – опорні шарніри.

Ця класифікація є однією із ключових для розуміння принципів роботи різних МПС від простих біподів до багатоланкових гексаподів [40, 42].



а) б)

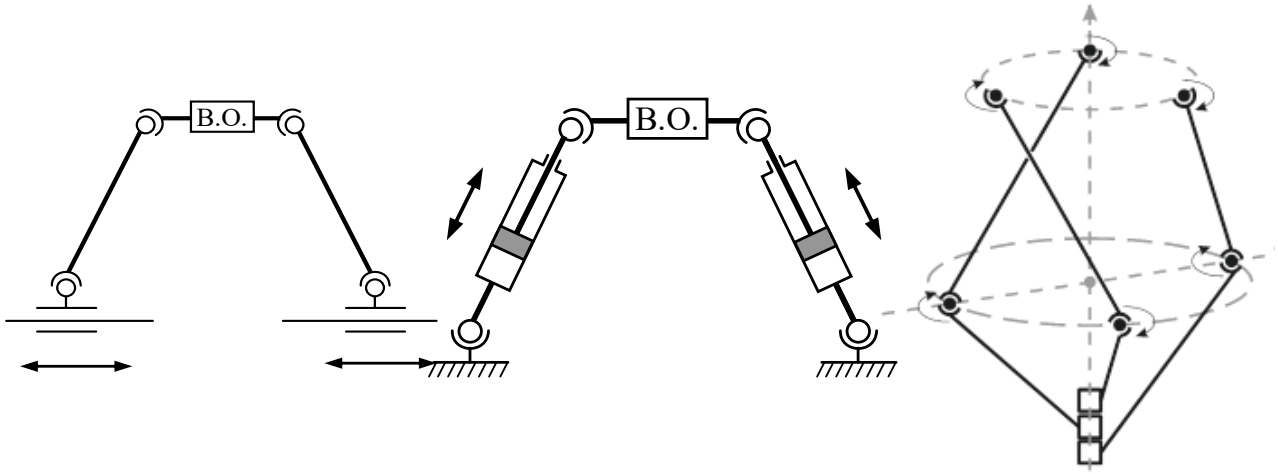
Рис. 1.2 – Механізми з кінематичними ланками: а) постійної довжини, б) змінної довжини

1.1.2.3. Класифікація за типом кінематичних пар і шарнірів

Для МПС характерним є широке застосування спеціальних кінематичних пар, що дозволяють забезпечувати необхідні ступені свободи, серед яких виділяються: опорний шарнір, сферичний шарнір, універсальний шарнір, шарніри у замкнених і незамкнених кінематичних ланцюгах, паралельні та послідовні з'єднання механізмів, сферичні МПС, осі яких перетинаються в одній точці. Ці елементи визначають кінематичну топологію механізму та його можливості щодо реалізації складних просторових рухів [40, 42].

1.1.2.4. Класифікація за кінематичною топологією

За кінематичною топологією виділяють структурні типи МПС, що формують основу їх кінематичної топології: біглайд (рис. 1.3 а) – механізм, з паралельною кінематикою, побудований на основі кінематичного з'єднання двох штанг змінної довжини; біпод (рис. 1.3 б) – механізм, з паралельною кінематикою, побудований на основі кінематичного з'єднання двох штанг постійної довжини; сферичний механізм паралельної структури (рис. 1.3 в) – механізм, у якому всі постійні та миттєві осі обертання ланок перетинаються в одній точці. Це окремі топологічні класи, що визначають характер руху платформи – поступальний, сферичний, комбінований тощо [40, 42].



а) б) в)

Рис. 1.3 – топологічні класи МПС: а) біглайд, б) біпод, в) сферичний механізм паралельної структури

1.1.2.5. Класифікація за розташуванням приводу та виконавчого органу

Механізми з паралельною структурою відрізняються також за місцем розташування приводів і ВО відносно бази механізму, виділяються такі варіанти: привід розташований на рухомому виконавчому органі; привід розташований на нерухомій частині механізму; комбіноване розташування додаткових приводів.

Окремо наведено класифікацію за просторовим розташуванням ВО відносно об’єкта позиціонування (рис. 1.4) – над об’єктом, під об’єктом, збоку тощо [40, 42].

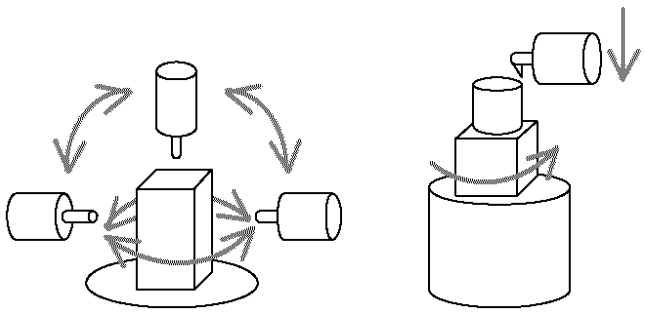


Рис. 1.4 – Розташування ВО по відношенню до об‘єкту позиціонування

1.1.3. Галузі використання механізмів паралельної структури

Огляд літератури свідчить про надзвичайно широку сферу застосування МПС в широкому спектрі високоточних промислових і науково-технічних

застосувань У промисловості МПС використовують там, де потрібні висока точність, жорсткість, швидкість реакції або здатність працювати з великими силами. Саме ці властивості обумовлюють широке використання механізмів паралельної структури в авіації, медицині, транспортному машинобудуванні, робототехніці, виробництві електроніки, вимірювальних комплексах, альтернативній енергетиці та сучасних системах орієнтації, зокрема в сонячних трекерах [1, 2, 4, 5, 28, 48].

1.1.3.1. Аерокосмічна техніка

В аерокосмічній техніці платформа Стюарта використовується як основа для систем моделювання рухів літальних апаратів, стабілізації антен, оптичних приладів і високоточних випробувальних стендів із шістьма ступенями свободи. Сучасні дослідження показують, що саме паралельна структура дозволяє відтворювати шість ступенів свободи з високою точністю та мінімальними затримками навіть у складних умовах динамічного навантаження [4, 5, 48].

Переналаштовувані паралельні платформи забезпечують необхідну точність і дозволяють відтворювати складні просторові траєкторії, що є ключовим для тестування авіаційних та космічних модулів і симуляторів руху, де необхідно забезпечує високу точність і можливість моделювання складних траєкторій, також і для систем навігаційного калібрування [5, 51].

1.1.3.2. Транспортні засоби та сільськогосподарська техніка

У транспортних засобах та сільськогосподарській техніці механізми паралельної структури застосовуються у стендах дорожніх випробувань, системах активної підвіски, платформах стабілізації кабін техніки та обладнання на рухомих машинах, включно з віброзахисними системами для вантажівок і комбайнів, що підтверджується оглядами гібридних паралельно-послідовних маніпуляторів [3, 49].

Актуальні роботи демонструють можливість комбінувати паралельну структуру із гідравлічними приводами для підвищення надійності та зменшення спотворень при великих навантаженнях. Завдяки підвищеній жорсткості та

високій динамічній чутливості МПС забезпечують стабільність і точність роботи обладнання в умовах значних вібраційних навантажень [49].

1.1.3.3. Медичне обладнання та електронна промисловість

У медичному обладнанні та електронній промисловості механізми паралельної структури застосовуються для мікропозиціонування та забезпечення плавності рухів хірургічних інструментів, у роботизованих системах радіохірургії, лабораторній робототехніці та високоточних оптичних комплексах [4, 6, 48].

У виробництві електроніки Delta-роботи є стандартом для високошвидкісного SMT-монтажу, а 3-DOF і 6-DOF паралельні маніпулятори забезпечують точність у мікрометровому діапазоні, що є критично важливим для нанотехнологічних та монтажних процесів. А також підтверджується численними оглядами та аналітичними роботами з індустріального застосування МПС [2, 4, 38].

1.1.3.4. Робототехнічні та вимірювальні системи

У робототехнічних та вимірювальних системах МПС використовуються завдяки високій повторюваності та мінімальним деформаціям, що робить їх незамінними в координатно-вимірювальних машинах, калібрувальних комплексах, лабораторних маніпуляторах та високоточних робототехнічних системах [39, 40]. Завдяки однорідній жорсткості та низьким похибкам МПС забезпечують стабільне та точне позиціювання в усьому робочому об’ємі, що є критично важливим для метрологічних та дослідницьких задач. Новітні методи оптимізації робочого простору, запобігання сингулярностям та структурного синтезу наведені в сучасних дослідженнях [43-45].

1.1.3.5. Спортивні тренажери та індустрія розваг

У спортивних тренажерах та індустрії розваг механізми паралельної структури, зокрема гексаподні платформи, використовуються для створення систем повного занурення, здатних у реальному часі відтворювати складні рухи

та прискорення. Платформи Стюарта стали стандартом у тренажерах для авто-, авіа- та мотоспорту, а також у VR-системах, оскільки забезпечують шість ступенів свободи та високу реалістичність руху. Що підтверджується як класичними робототехнічними оглядами, так і новими дослідженнями з динаміки гібридних МПС [4, 5, 47, 48].

1.1.3.6. Сонячні трекери

У сонячних трекерах паралельна структура механізму використовується завдяки здатності забезпечувати плавне та точне двовісне орієнтування панелей, включно з компенсацією вітрових навантажень і стабільним дотриманням заданого кута наведення. Оскільки основною функцією сонячного трекера є просторове позиціонування робочої поверхні з необхідною точністю орієнтації, його кінематичну схему доцільно розглядати в межах теорії механізмів паралельної структури. Такий підхід дає можливість застосовувати апарат прямих і зворотних кінематичних перетворень, ураховувати реальні конструктивні особливості механізму та будувати математичні моделі, придатні для підвищення точності наведення й компенсації похибок у процесі експлуатації [28, 38, 40, 52].

1.2. Методи математичного моделювання маніпуляторів

Для визначення способу спрощення розрахунку керуючих впливів необхідно розглянути існуючі методи математичного моделювання просторових механізмів. Застосування методів математичного програмування та наближення функцій потребує формалізації функціональних зв'язків між внутрішніми та вихідними параметрами синтезу, які визначаються математичними моделями [50].

Побудова математичних моделей руху маніпуляторів і елементів МПС ґрунтується на декількох фундаментальних підходах. Найпоширенішими є метод замкнутого векторного контуру та метод перетворення координат. Обидва ці методи докладно описані у [27] та широко застосовуються в сучасних

робототехнічних системах, зокрема в механізмах паралельної структури та сферичних паралельних механізмах.

Методи мають різні області використання, зокрема, метод замкнутого векторного контуру призначений для моделювання плоских або близьких до плоских механізмів, а в той час як метод перетворення координат застосовується для просторових багатоланкових структур, у тому числі платформ Стюарта та гібридних паралельно-послідовних механізмів [23, 27, 50, 53].

1.2.1. Метод замкнутого векторного контуру

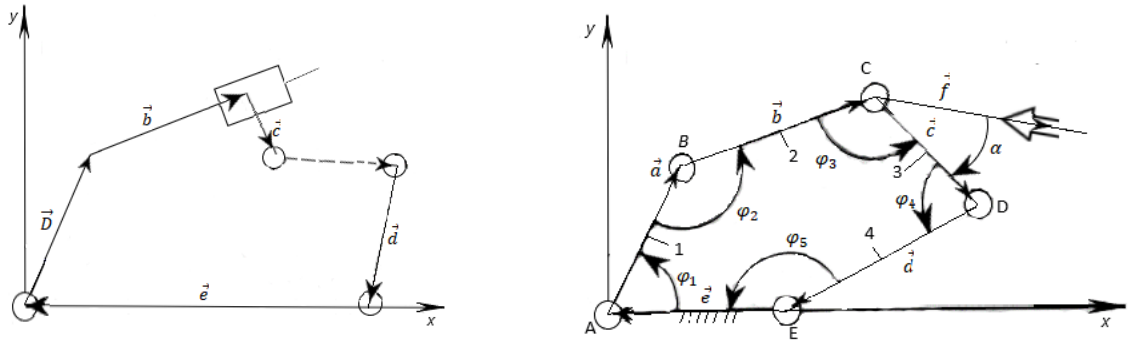
Метод замкнутого векторного контуру. Даний метод застосовується для моделювання плоских механізмів і полягає в тому, що їх ланки та напрямні поступальних кінематичних пар (КП) є відповідними векторами, що утворюють замкнутий контур (рис. 1.5 а), сума векторів якого визначається виразом:

$$\bar{a} + \bar{b} + \bar{c} + \cdots + \bar{d} + \bar{e} = 0. \qquad (1.1)$$

Для плоских механізмів сума векторів ланок та напрямних поступальних кінематичних пар дорівнює нулю. Це дозволяє розкласти рівняння на скалярні компоненти за осями x та y, що дає систему тригонометричних рівнянь для визначення координат точок та кутових положень ланок [27, 53].

Застосування методу включає побудову рівнянь руху п’ятизвенних плоских механізмів (рис. 1.5 а), опис кінематики механізмів автооператорів (рис. 1.5 б), а також формування геометричних залежностей без застосування складних матричних методів.

Метод забезпечує просте й інтуїтивне моделювання плоских механізмів, дозволяє отримувати залежності для положення рухомих точок та визначати керуючі параметри для приводів, але принциповим обмеженням методу замкнутого векторного контуру є придатність лише для плоских механізмів, оскільки не враховуються просторові повороти та 3D-орієнтацію ланок [27].



а) б)

Рис. 1.5 – Побудова замкнутого векторного контуру: а) шарнірний п'ятиланковий механізм, б) механізм автооператора

У сучасній літературі метод векторного контуру використовується для аналізу планарних маніпуляторів і n-ланкових механізмів, оцінки точності планарних механізмів на основі комплексного запису векторної петлі, моделювання кабельних та комплаєнтних механізмів, де довжини тросів утворюють замкнені контури [50].

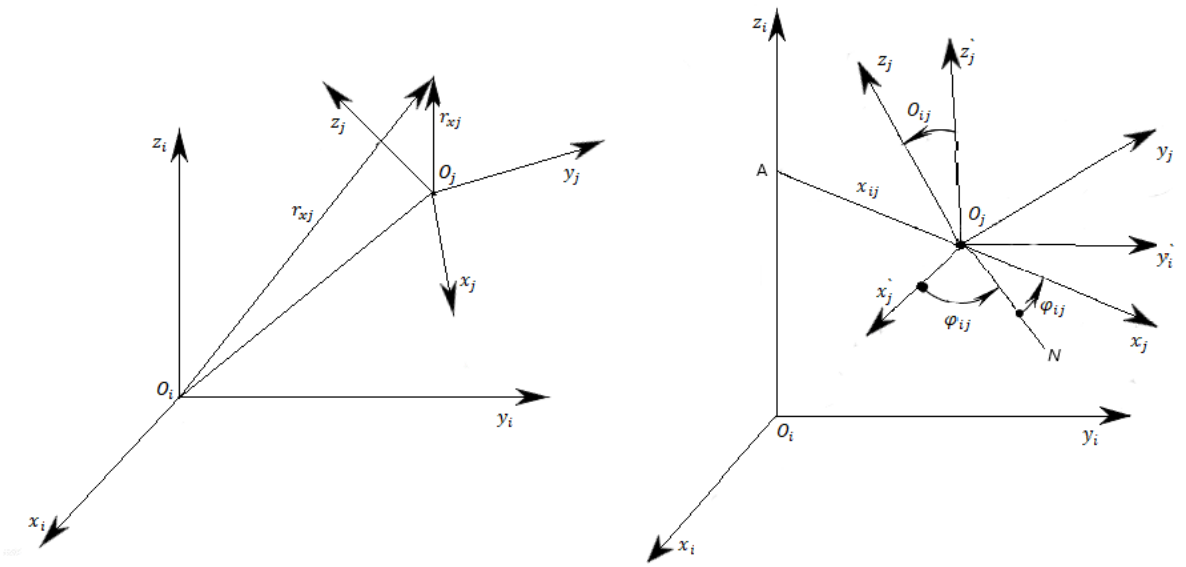
Таким чином, метод замкнутого векторного контуру залишається одним із найефективніших аналітичних інструментів для задач плоскої кінематики, а також часто виступає першим етапом до побудови повної 3D-моделі механізму [50, 53].

1.2.2. Метод перетворення координат

Метод перетворення координат. Одним з найбільш зручних методів, які використовуються для складання рівнянь схем просторових механізмів, є метод, заснований на використанні матриць перетворення координат четвертого порядку та дуальних матриць, що забезпечують опис поворотів і перенесень у єдиному математичному форматі [23, 27, 50]. Нехай з ланками і та j, які утворюють кінематичну пару, зв’язані системи координат $T_i(O_i x_i y_i z_i)$ та $T_j(O_j x_j y_j z_j)$ (рис. 1.6 а). Тоді перетворення системи T_j в T_i здійснюється матричним рівнянням:

$$r_{ki} = M_{ij} r_{kj}, \tag{1.2}$$

де r_{ki} – матриця-стовпець координат точки К в системі T_i; де r_{kj} – матриця-стовпець координат точки К в системі T_j.



а) б)

Рис. 1.6 – Схеми: а) визначення матриць перетворення координат, б) вираження напрямних косинусів через кути Ейлера

Блочна матриця перетворення має вигляд:

$$M_{ij} = \begin{bmatrix} L_{ij} & r_{0ij} \\ 0 & 1 \end{bmatrix}, \tag{1.3}$$

де $L_{ij} = [a_{ij}]$; $r_{0ij} = [x_{0ij},\ y_{0ij},\ z_{0ij}]^T$ – вектор-стовпець зміщень початку координат системи T_j відносно T_i; $i = \overline{1,3}$; $j = \overline{1,3}$ – ортогональна матриця третього порядку, елементами якого є направляючі косинуси тобто:

$$L_{ij} = \begin{Vmatrix} \cos(\widehat{x_i, x_j}) * \cos(\widehat{x_i, y_j}) * \cos(\widehat{x_i, z_j}) \\ \cos(\widehat{y_i, x_j}) * \cos(\widehat{y_i, y_j}) * \cos(\widehat{y_i, z_j}) \\ \cos(\widehat{z_i, x_j}) * \cos(\widehat{z_i, y_j}) * \cos(\widehat{z_i, z_j}) \end{Vmatrix}, \tag{1.4}$$

де x_{0ij}, y_{0ij}, z_{0ij} – координати початку O_j системи координат T_j в системі координат T_i (рис. 1.6 б).

Направляючі косинуси в матриці L_{ij} виражаються через кути Ейлера, для відліку яких спочатку відмічаються лінії вузлів.

Метод перетворення координат є універсальним підходом для моделювання просторових механізмів, у яких положення і орієнтація кожної ланки описуються відносно локальних систем координат [23, 27, 50]. Перевагами методу є можливість моделювати будь-які 3D-переміщення у складних маніпуляторах; формувати прямі та обернені кінематичні рівняння МПС; описувати багатоланкові просторові структури незалежно від кількості ступенів свободи; будувати повні кінематичні моделі у вигляді добутку матриць:

$$M = M_0^1 \; M_1^2 \ldots M_{n-1}^n \tag{1.5}$$

У міжнародній практиці метод координатних перетворень найчастіше реалізується через Denavit–Hartenberg (DH) або модифіковану MDH-конвенцію, де кожна ланка описується чотирма параметрами. DH-метод є стандартом у побудові кінематики серійних та гібридних маніпуляторів і широко використовується у сучасних дослідженнях. DH-параметри дозволяють: уніфікувати побудову матриць перетворень; зменшити кількість аналітичних операцій; адаптувати метод до аналізу окремих гілок МПС у складі паралельно-послідовних структур [38, 42, 50].

1.3. Технічні засоби реалізації систем керування

Ефективність систем керування робототехнічними механізмами значною мірою визначається не лише алгоритмами керування, але й апаратними засобами, які забезпечують збір даних, їх обробку та формування керуючих сигналів. У випадку сферичних паралельних механізмів технічна реалізація системи керування включає комплекс апаратних компонентів, що забезпечують взаємодію

між сенсорними підсистемами, обчислювальними модулями та виконавчими приводами [19, 25].

Типова структура апаратної частини системи керування складається з декількох основних елементів: сенсорних пристроїв, обчислювального модуля, інтерфейсів зв’язку та виконавчих механізмів. Сенсорні підсистеми забезпечують вимірювання параметрів руху механізму, таких як положення, швидкість, прискорення або орієнтація платформи. Для цього можуть використовуватися різні типи датчиків, зокрема енкодери, інерціальні вимірювальні модулі (IBM), гіроскопи, акселерометри та оптичні системи відстеження.

Зібрані сенсорні дані передаються до обчислювального модуля, який виконує обробку інформації, оцінку стану системи та формування керуючих сигналів. У сучасних робототехнічних системах для цих задач застосовуються мікроконтролери, одноплатні комп’ютери або спеціалізовані вбудовані системи реального часу. Такі пристрої забезпечують виконання алгоритмів кінематики, динаміки, фільтрації сигналів та систем керування у режимі реального часу.

Окрему роль відіграють інтерфейси зв’язку, які забезпечують обмін даними між компонентами системи. Залежно від вимог до швидкодії та надійності можуть використовуватися різні протоколи передачі даних, зокрема SPI, I2C, UART, CAN або Ethernet. Використання промислових шин зв’язку дозволяє підвищити завадостійкість системи та забезпечити синхронізовану роботу всіх елементів керування.

Виконавча частина системи керування представлена приводами, які безпосередньо забезпечують рух механізму. У сферичних паралельних роботах зазвичай застосовуються електричні серводвигуни або крокові двигуни з відповідними драйверами керування. Драйвери перетворюють керуючі сигнали обчислювального модуля у сигнали живлення двигунів, що дозволяє реалізувати необхідні траєкторії руху платформи.

Сучасні системи керування все частіше будуються за принципом модульної архітектури, у якій сенсорні модулі, обчислювальні пристрої та виконавчі механізми можуть легко інтегруватися в єдину систему. Такий підхід забезпечує

гнучкість конфігурації системи, спрощує модернізацію апаратної частини та дозволяє адаптувати систему керування до різних робототехнічних застосувань.

Розглянемо основні технічні засоби, що застосовуються для реалізації систем керування сферичними паралельними механізмами, зокрема сенсорні системи вимірювання положення та орієнтації, обчислювальні платформи реального часу, а також виконавчі приводи та інтерфейси зв’язку.

1.3.1. Сенсорні системи вимірювання положення та орієнтації

У залежності від вимог до точності вимірювань, умов експлуатації та характеру контрольованих параметрів у подібних системах може використовуватися широкий спектр різних типів датчиків [13], зокрема:

1. Енкодери забезпечують вимірювання кутового положення валів серводвигунів, що дозволяє визначати поточний стан приводів з високою точністю. Вони є базовим засобом зворотного зв'язку в системах керування приводами. (рис. 1.7)

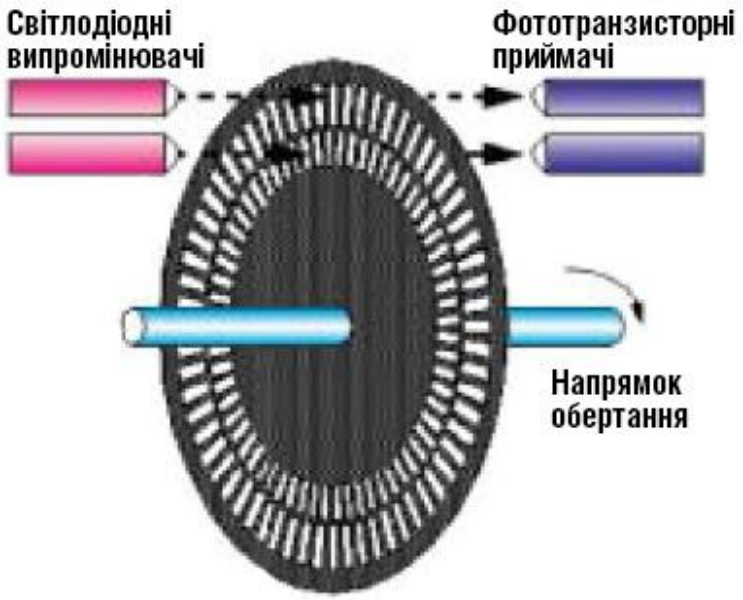


Рис. 1.7– Схема енкодера

2. Ультразвукові далекоміри відносяться до активного типу пристроїв вимірювання відстані [54]. Принцип їх роботи полягає у випромінюванні звукового сигналу, що знаходиться за межами діапазону, який здатна чути

людина, та вимірюванні часу його відбиття від об'єкта. Детальна схема далекоміру зображена на рис. 1.8.

До переваг можна віднести відносно невелику ціну в порівнянні з іншими модулями. Але його недоліки такі як невисока точність вимірювання, залежність від середовища, залежність від погодних умов, обмеженість по відстані виміру ($min = 0{,}3$м, а $max = 20$м) та особливості поширення звукової хвилі, яка є наслідком нелінійності поширення сигналу, що призводить до ускладнень при вимірюванні ультразвуковими далекомірами кутів та діагоналей.

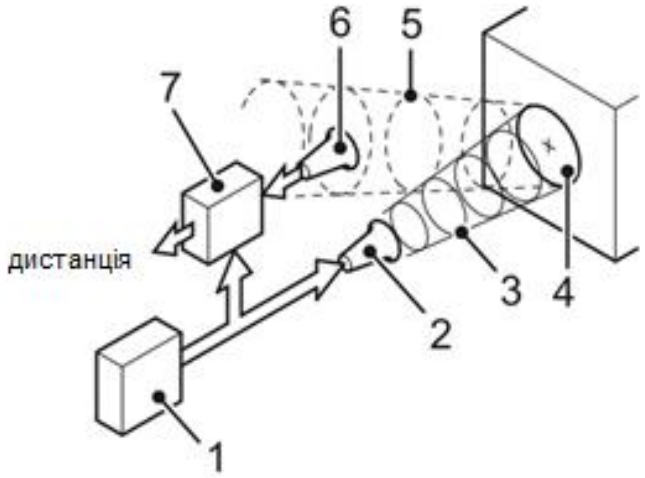


Рис. 1.8– Схема ультразвукового далекоміру:
1 – генератор, 2 – ультразвуковий передатчик, 3 – звукові імпульси, що передаються, 4 – вимірюваний об’єкт, 5 – відбиті хвилі, 6 – ультразвуковий приймач, 7 – обчислювач

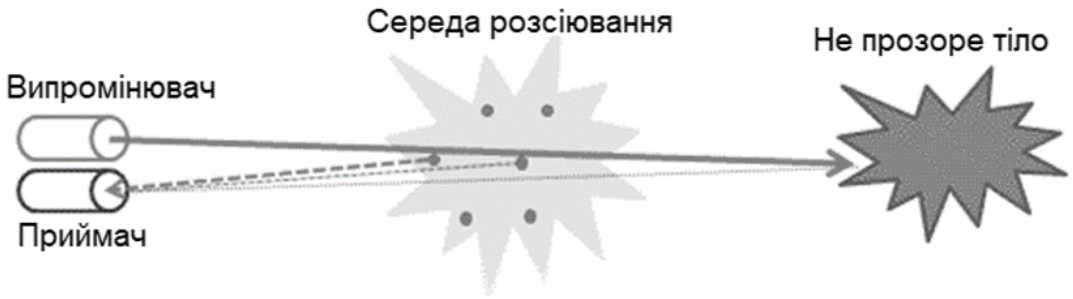
3. Лазерні далекоміри забезпечують вимірювання відстаней із застосуванням лазерного променю [55]. Розрізняють імпульсні (на основі визначення часу проходження сигналу до об'єкта і назад) та фазові (на основі різниці фаз світлового сигналу).

Принцип роботи лазерних далекомірів наведено на рис. 1.9.

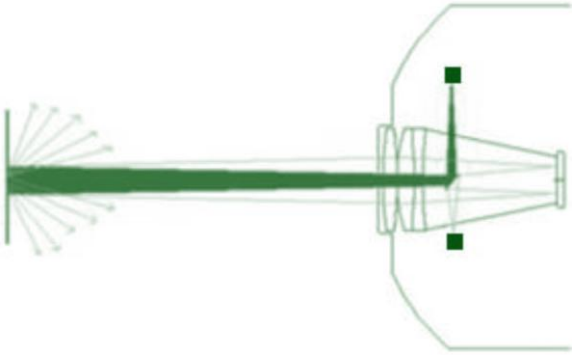
Перевагою є великий діапазон відстані вимірювання ($min = 0{,}02 - 0{,}03$м, а $max = 100 - 200$м), а до недоліків, можна віднести, такі як

залежність від середовища, залежність від погодних умов та особливості поширення світової хвилі.

4. Радіодалекомір – це пристрій для визначення відстані безконтактним методом за допомогою радіохвиль, технічно реалізоване у вигляді автономного приладу або в складі радіо далекомірної системи. Радіодалекомір працює на основі принципів радіолокації. На даний час використовується лише у вигляді матриці радіодалекомірів – радар. Має ті ж недоліки, що й ультразвуковий далекомір [56].



а)



б)

Рис. 1.9 – Принцип роботи лазерних далекомірів: а) Схема імпульсного лазерного далекоміру, б) Оптична схема фазового далекоміру

5. Найбільш ефективними для задач керування сферичними паралельними механізмами є модулі акселерометра та гіроскопа (рис. 1.10). Акселерометри використовуються для вимірювання статичного гравітаційного прискорення, що дозволяє визначати кут відхилення вимірюваного об'єкта від вертикалі, а також дозволяє вимірювати динамічне прискорення, викликане поштовхами, рухами та ударами або вібраціями, тобто коливаннями, які мають

малу амплітуду та низьку частоту, яка знаходиться в діапазоні декількох десятків Герц.

Принцип дії акселерометра полягає в тому, що сила прискорення вимірюється в одиницях g і може вимірюватися в одній, двох або трьох площинах. Зараз найчастіше використовуються 3-осьові акселерометри, конструкція яких складається з системи трьох акселерометрів, і кожен з них вимірює прискорення в іншому напрямку – в площинах X, Y і Z. Прикладом 3-осьового акселерометра може бути модель GY-521 [57]. Але є й 6-осьові акселерометри, або ж навіть з 9- та 10-ма ступенями свободи, але це досягаються вже за рахунок додавання інших датчиків, на кшталт магнітометра, датчика тиску та інших.

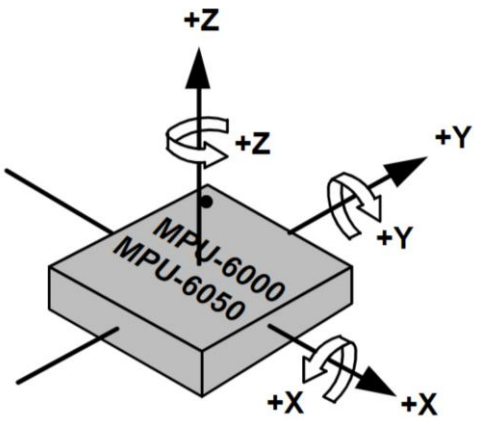


Рис. 1.10 – Орієнтація осей чутливості та полярність обертання акселерометра

Якщо прискорення у будь-якій з площин діє в протилежному напрямку, від напрямку в якому спрямована вісь датчика, акселерометр вимірює прискорення з від'ємним значенням. Якщо ж напрямок прискорення збігається з напрямком осі датчика, то отримуємо значення з позитивним знаком.

Якщо на акселерометр не відбувається вплив якогось зовнішнього прискорення, то пристрій вимірюватиме лише прискорення землі, тобто силу гравітації. Якщо припустити, що 3-осьовий акселерометр, розміщений таким чином, що датчик на осі X спрямований вліво, датчик на осі Y вниз, а датчик на осі Z вперед, і на нього не впливають ніякі сили, тоді акселерометр поверне значення: $X = 0\,g,\ Y = 1\,g,\ Z = 0\,g$. Якщо цей же акселерометр буде

відхилений вліво, його показання будуть: $X = 1\,g$, $Y = 0\,g$, $Z = 0\,g$. Аналогічно, якщо відхилення відбудеться вправо, площина X поверне результат $X = -1\,g$. Наведені залежності вимірювання прискорення використовуються алгоритмами систем, що контролюють роботу акселерометра.

Серед основних типів акселерометрів можна виділити три типи: ємнісні акселерометри MEMS, п'єзоелектричні акселерометри та п'єзорезистивні акселерометри.

Ємнісні акселерометри MEMS. Ємнісні акселерометри, які використовують технологію MEMS [13, 58], – це найдешевші, найпоширеніші та найменші акселерометри на ринку. Принцип його роботи полягає в розміщенні вантажу, встановленого на пружинах. Один кінець пружини прикріплений до обкладок гребінчастого конденсатора, а інший кінець – до закріпленого вантажу. Під впливом сили, яка діє на датчик, вантаж рухається на пружинах, що призводить до зміни відстані між конденсаторним елементом і масою, і тим самим впливає на зміну ємності.

Застосовуються, в переносних пристроях, мобільних приладах та широко поширеній побутовій електроніці. Однією з найбільших переваг є можливість їх установки безпосередньо на друкованій платі. До недоліків слід віднести низьку точність вимірювань, особливо у разі вимірювань вищих амплітуд і частот, що робить їх непридатними для спеціалізованих промислових застосувань.

П'єзорезистивні акселерометри. Наступним видом акселерометрів є датчики, що використовують п’єзорезистивний ефект. Принцип дії схожий на дію тензометра, тобто датчика, що вимірює напругу. Цей тип акселерометра оснащений п'єзорезистивним матеріалом, який деформується під впливом зовнішньої сили, приводячи до зміни опору. Далі зміна опору перетворюється в електричний сигнал, що отримується через об’єднаний з акселерометром приймач. Їх превагою є великий діапазон вимірювань, можливістю виміряти повільно змінювані сигнали. Недоліком є нечутливість до змін температури навколишнього середовища, проблеми з виявленням слабких сигналів, а також вища ціна в порівнянні з ємнісними акселерометрами MEMS.

П'єзоелектричні акселерометри. Цей акселерометр є одним із найбільш часто використовуваних датчиків для вимірювання рівня вібрації. З цієї ж причини п'єзоелектричні акселерометри зазвичай використовуються в промисловості для діагностики та контролю машин і пристроїв. Його робота схожа на принцип дії п'єзорезистивних систем. Однак під впливом прискорення вони не змінюють свого опору, а генерують електричну напругу певного значення. Вимірювальним елементом цих датчиків зазвичай є цирконат-титанат свинцю. Цирконат-титанат свинцю, деформуючись, виробляє електричний заряд. П'єзоелектричні акселерометри відрізняються високою чутливістю і точністю, завдяки чому вони знаходять застосування в багатьох галузях – від надзвичайно просунутих і точних сейсмічних вимірювань до ударних тестів та руйнівних випробувань, що проводяться в несприятливих умовах. Вихідний сигнал п'єзоелектричних акселерометрів зазвичай піддається посиленню і температурній компенсації. Розрахунку переміщення об'єкта сприяє пересилання сигналу на вихід інтегратора.

Інші акселерометри. Серед інших конструкцій акселерометра слід перелічити конструкції IEPE, які повсюдно використовуються для вимірювання вібрацій. Варто також згадати п'єзоелектричні зарядні акселерометри, придатні для роботи при екстремальних температурах [58].

1.3.2. Обчислювальні платформи реального часу

Зібрані сенсорні дані передаються до обчислювального модуля, який виконує обробку інформації, оцінку стану системи та формування керуючих сигналів. У сучасних робототехнічних системах для цих задач застосовуються мікроконтролери, одноплатні комп'ютери або спеціалізовані вбудовані системи реального часу [19, 59]. Такі пристрої забезпечують виконання алгоритмів кінематики, динаміки, фільтрації сигналів та систем керування у режимі реального часу.

Вибір конкретної обчислювальної платформи залежить від вимог до швидкодії, точності обчислень та енергоспоживання. Мікроконтролери (Arduino, STM32, ESP32, ESP8266) забезпечують низьку затримку та детерміновану

поведінку для керування приводами [60-62]. Одноплатні комп'ютери (Raspberry Pi, NVIDIA Jetson) дозволяють виконувати складні обчислення для обробки відеоданих та реалізації алгоритмів комп'ютерного зору [63, 64].

1.3.3. Виконавчі приводи та інтерфейси зв'язку

Виконавча частина системи керування представлена приводами, які безпосередньо забезпечують рух механізму. У сферичних паралельних механізмах зазвичай застосовуються електричні серводвигуни або крокові двигуни з відповідними драйверами керування [19, 65]. Драйвери перетворюють керуючі сигнали обчислювального модуля у сигнали живлення двигунів, що дозволяє реалізувати необхідні траєкторії руху платформи.

Окрему роль відіграють інтерфейси зв'язку, які забезпечують обмін даними між компонентами системи [66-69]. Залежно від вимог до швидкодії та надійності можуть використовуватися різні протоколи передачі даних, зокрема SPI, I2C, UART, CAN або Ethernet, а також промислові протоколи, такі як Modbus (RTU/TCP), PROFIBUS, PROFINET, EtherCAT або DeviceNet. Використання промислових шин зв'язку дозволяє підвищити завадостійкість системи та забезпечити синхронізовану роботу всіх елементів керування.

Сучасні системи керування все частіше будуються за принципом модульної архітектури, у якій сенсорні модулі, обчислювальні пристрої та виконавчі механізми можуть легко інтегруватися в єдину систему. Такий підхід забезпечує гнучкість конфігурації системи, спрощує модернізацію апаратної частини та дозволяє адаптувати систему керування до різних робототехнічних застосувань.

1.4. Методи ідентифікації параметрів руху МПС

Високоточна ідентифікація параметрів руху механізмів паралельної структури, зокрема сферичних паралельних механізмів, є критично важливою для забезпечення точності позиціонування, стабільності керування та ефективної компенсації зовнішніх збурень [1, 6, 25]. Системи такого класу характеризуються високою динамікою, багатовимірними просторовими траєкторіями та значною

чутливістю до похибок вимірювання, що унеможливлює використання лише одного типу сенсорів без додаткової обробки даних.

У сучасних дослідженнях і практичних реалізаціях застосовується сукупність методів ідентифікації, які базуються на оптичних, інерціальних, магнітних сенсорах, а також на алгоритмах злиття даних і модельно-орієнтованих підходах, що дозволяють досягти високої точності та стійкості відстеження руху [1, 6, 71].

Нижче наведено основні групи методів ідентифікації руху, які можуть бути застосовані для СПМ.

1.4.1. Оптичне відстеження

Оптичні методи засновані на використанні відеокамер, систем комп'ютерного зору та алгоритмів розпізнавання положення об'єктів [20, 21]. Сучасні методи машинного зору дозволяють отримувати прецизійні дані про положення та орієнтацію рухомої платформи СПМ у режимі реального часу.

У сучасних дослідженнях все ширше застосовуються алгоритми машинного навчання, такі як Google Mediapipe [72, 73], які здатні автоматично визначати ключові точки та просторові орієнтації об'єктів. Наприклад, у роботах, що моделюють рухи голови та очей у біомеханічних системах керування, використовувалася система Mediapipe для визначення орієнтації обличчя, яку далі відтворював 3-DOF сферичний паралельний робот.

До переваг методу оптичного відстеження відноситься висока точність позиційної оцінки, можливість відстеження складних рухів без фізичного контакту, інтеграція з глибинними нейронними мережами.

В той час як недоліками є залежність від освітлення та умов навколишнього середовища, можливі затримки у випадку складної обробки зображення.

1.4.2. Інерціальне відстеження

Інерціальні вимірювальні модулі складаються з акселерометрів, гіроскопів та магнітометрів, що дозволяють вимірювати лінійні прискорення, кутові швидкості та абсолютну орієнтацію [13, 58]. Дані з IBM інтегруються у часі для

отримання траєкторії руху, що робить цей метод надзвичайно швидким і незалежним від зовнішніх умов.

Основними перевагами методу є: висока частота оновлення (до сотень Гц); низька затримка, що важливо для реального часу; нечутливість до освітлення та зовнішніх оптичних перешкод. Серед недоліків слід зазначити: накопичення помилки через дрейф гіроскопів; потреба у корекції або обмеження часу інтегрування.

1.4.3. Магнітне відстеження

Магнітні методи відстеження використовують магнітометри для визначення орієнтації рухомої платформи СПМ відносно магнітного поля Землі або штучних магнітних маркерів [74]. Цей підхід часто застосовується як доповнення до інерціальних сенсорів, оскільки дозволяє компенсувати дрейф гіроскопів та стабілізувати орієнтацію.

Основними перевагами методу є: абсолютне визначення орієнтації без інтегрування; нечутливість до освітлення чи візуальних перешкод. Серед недоліків слід зазначити: чутливість до локальних магнітних спотворень (металеві конструкції, електродвигуни); нижча просторово-кутова точність порівняно з оптичними методами.

1.4.4. Візуально-інерціальне злиття даних

Візуально-інерціальні системи (ВІС) поєднують оптичні та інерціальні вимірювання, об’єднуючи їх у єдину модель через методи фільтрації (наприклад, фільтр Калмана) або методи оптимізації [75-86].

При такому способі компенсуються недоліки окремих методів (дрейф ІВМ та нестабільність оптики), забезпечується висока точність і стійкість у реальному часі і здатність працювати навіть при часткових втратах даних.

Застосування ВІС є особливо ефективним у СПМ, де необхідно поєднати високу частоту оновлення, яку забезпечує ІВМ, з просторовою точністю, яку дає оптичний модуль.

1.4.5. Модельно-орієнтоване відстеження

Модельно-орієнтовані методи ґрунтуються на використанні аналітичних або чисельних математичних моделей СПМ для визначення руху [27, 50]. Вони враховують кінематичні рівняння, геометричні залежності та динамічні властивості платформи.

Прикладами таких методів є: регресори Slotine–Li, які застосовуються для ідентифікації динамічних параметрів (інерційні моменти, маси ланок, коефіцієнти тертя); чисельні алгоритми розв’язання оберненої кінематики; моделі стану системи для прогнозування руху.

Оскільки забезпечується відсутність залежності від шумів сенсорів, можливість прогнозування поведінки механізму та підвищення точності через використання структурних властивостей СПМ – цей метод відстеження є одним з найпоширеніших. Проте має певні недоліки, зокрема критичність до правильності математичної моделі та значна обчислювальна складність у механізмах з високою рухомістю.

Таким чином, методи ідентифікації руху СПМ охоплюють різні за природою підходи — від оптичних систем до інерціальних, магнітних, комбінованих та модельно-орієнтованих. У реальних застосуваннях найефективнішими є гібридні методи, які поєднують сенсорні дані та математичні моделі, забезпечуючи одночасно високу точність, швидкодію та стійкість до зовнішніх впливів.

1.5. Висновки до розділу 1

Сферичні паралельні механізми належать до перспективного класу механічних систем, які поєднують переваги паралельної кінематичної структури з можливістю реалізації просторових обертальних рухів навколо фіксованого центра. Завдяки високій жорсткості конструкції, підвищеній точності позиціонування, здатності сприймати значні навантаження та компактності такі механізми знаходять застосування в робототехніці, і багатьох інших галузях, де потрібно забезпечити високу точність орієнтації об'єктів у просторі.

Значна кількість архітектур СПМ, що відрізняються типами кінематичних пар, кількістю ступенів свободи, конфігурацією ланок і робочими характеристиками з одного боку розширює можливості їх практичного застосування, а з іншого — ускладнює процес аналізу, синтезу та вибору оптимальної структури для конкретних умов експлуатації.

З появою доступних обчислювальних засобів, сучасних методів математичного моделювання та ідентифікації, програмно-технічних засобів вимірювання і автоматизованого проєктування, а також високоточних систем керування, з’явилася можливість реалізувати переваги паралельних механізмів на практиці, що сприяє інтенсивному розвитку цього напряму.

Для побудови математичних моделей просторових механізмів застосовуються методи замкнутого векторного контуру, перетворення координат, метод формально-фактичних параметрів, метод R-функцій тощо. Метод замкнутого векторного контуру застосовується для моделювання плоских механізмів і полягає в тому, що ланки та напрямні кінематичних пар є векторами, що утворюють замкнутий контур. Однак цей метод підходить лише для пласких механізмів. Метод перетворення координат, заснований на використанні матриць перетворення четвертого порядку та дуальних матриць, дозволяє складати рівняння просторових механізмів, але потребує специфічних математичних знань і складний для програмної реалізації. Тому доцільним є використання підходу, що ґрунтується на геометричному описі просторового положення елементів механізму та безпосередньому визначенні їх кінематичних параметрів. Такий підхід забезпечує наочність математичної моделі, спрощує програмну реалізацію алгоритмів і дозволяє враховувати реальні конструктивні особливості механізму.

Важливим напрямом досліджень СПМ є визначення та ідентифікація параметрів їх руху на основі даних, отриманих від технічних засобів вимірювання. Їх застосування дає змогу отримувати інформацію про положення, орієнтацію, швидкість та прискорення елементів механізму як у статичних, так і в динамічних режимах роботи.

Однак результати вимірювань зазвичай містять шуми, систематичні похибки та випадкові збурення, що обумовлює необхідність застосування математичних методів обробки та ідентифікації. Для підвищення точності оцінювання параметрів руху доцільно використовувати методи фільтрації та об'єднання даних від різних сенсорів. Поєднання вимірювальних засобів із математичними методами ідентифікації дозволить отримувати більш достовірні оцінки стану механізму порівняно з використанням окремих джерел інформації.

Особливої актуальності такі підходи набувають для СПМ, де навіть незначні похибки визначення кутового положення можуть призводити до суттєвих відхилень орієнтації робочого органу. Тому сучасні системи керування СПМ все частіше будуються на основі комплексного використання сенсорних даних та алгоритмів ідентифікації, що забезпечує підвищення точності позиціонування, стійкості до зовнішніх збурень і надійності функціонування механізму в реальних умовах експлуатації.

Попри наявність універсальних програмних пакетів для моделювання та керування робототехнічними системами, для сферичних паралельних механізмів доцільним є розроблення спеціалізованого програмного забезпечення, орієнтованого на особливості їхньої кінематичної структури. Такий підхід дає змогу спростити реалізацію алгоритмів керування, зменшити обчислювальну складність окремих процедур та підвищити ефективність використання обчислювальних ресурсів.

Розробка спеціалізованих програмних реалізацій дозволить враховувати характерні для СПМ особливості, зокрема обмеження робочої області, наявність множинних розв’язків зворотної задачі кінематики, сингулярні конфігурації, геометричні обмеження кінематичних ланцюгів та необхідність безперервного відстеження орієнтації рухомої платформи та інтегрувати до програмної системи специфічні алгоритми перевірки допустимості положень, вибору фізично реалізовуваного розв’язку та контролю кінематичних параметрів у реальному часі.

Спеціалізоване програмне забезпечення сприяє тісній інтеграції математичних моделей, вимірювальних засобів та алгоритмів ідентифікації

параметрів руху в межах єдиної системи керування, для підвищення точності позиціонування, спрощення налагодження механізму та адаптації програмних алгоритмів до конкретної конструкції СПМ.

Важливу роль у сучасних системах керування відіграють методи комп’ютерного зору та штучного інтелекту. Їх використання дає змогу автоматизувати процес визначення просторового положення об’єктів, оперативно аналізувати великі обсяги візуальної інформації та формувати керуючі впливи в реальному часі. Застосування алгоритмів машинного навчання та комп’ютерного зору надасть можливість скоротити час обробки даних, підвищити швидкодію системи та забезпечить більш ефективну реакцію на динамічні зміни зовнішнього середовища в задачах керуваннях сферичними паралельними механізмами.

Враховуючи переваги СПМ, такі механізми доцільно використовувати у задачах діагностики поверхонь технологічного обладнання, особливо під час контролю складнопрофільних об'єктів, де необхідно підтримувати заданий кут огляду, відстань до поверхні або траєкторію переміщення сенсора. Використання СПМ створить можливість інтеграції систем комп'ютерного зору, лазерного сканування та інших безконтактних методів контролю, що забезпечить автоматизацію процесу виявлення дефектів, оцінювання зношування та моніторингу технічного стану обладнання.

РОЗДІЛ 2 МАТЕМАТИЧНА МОДЕЛЬ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ

Сферичні паралельні механізми набули значної актуальності в сучасній робототехніці завдяки своїй винятковій універсальності та перевагам у різних галузях промисловості. Основними перевагами СПМ є висока точність позиціонування завдяки паралельній кінематичній структурі, підвищена мобільність, що дозволяє виконувати обертові та поступальні рухи в широкому діапазоні, довговічність та надійність конструкції, здатної витримувати значні навантаження, а також відносно мала вага завдяки використанню сучасних матеріалів [3, 4, 8, 11, 23, 35-37, 41, 42, 87, 88].

2.1. Опис прототипу сферичного паралельного механізму

На даний час для сферичного паралельного механізму досліджено робочий простір, закони переміщення кінематичних ланок матричними методами перетвореннями систем координат кожної ланки відносно іншої [3, 10, 12, 23, 26, 46, 47]. Однак даний метод дуже ускладнює наочне представлення комп'ютерними засобами, оцінку кінематичних параметрів таких механізмів на стадії проектування та моделювання.

З метою спрощення програмних процедур керування тривимірними проектами моделей пропонується використання законів аналітичної геометрії щодо визначення принципів керування сферичним і подібними йому механізмів для типових просторових траєкторій руху позиціонуючої платформи або об'єкту за яким виконується слідкування [26, 27].

Залежно від конкретного завдання та експлуатаційних вимог можуть застосовуватися різні методи керування СПМ: керування на основі приводів (електричних, гідравлічних або пневматичних), керування на основі сенсорів (енкодери, акселерометри, датчики сили), замкнене керування зі зворотним зв'язком та адаптивне керування [23]. В наслідок чого основна увага приділяється математичному моделюванню кінематики механізму як основі для реалізації

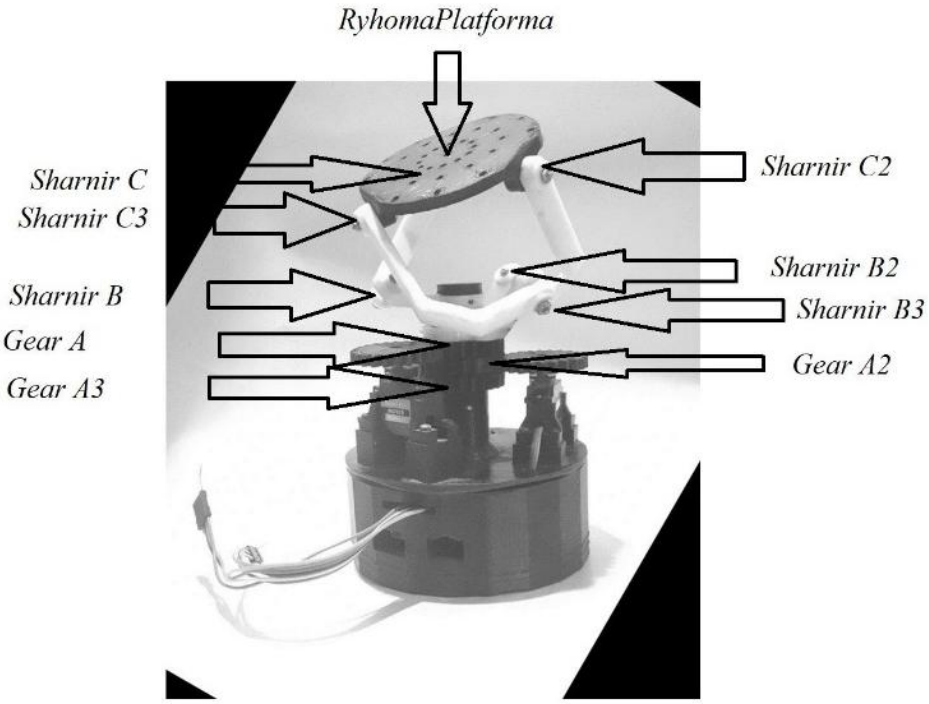
зазначених методів керування. Зовнішній вигляд прототипу СПМ виготовленого для дослідження та його кінематична модель наведені наведені на рис. 2.1.

Принцип роботи досліджуваного сферичного паралельного механізму базується на координованому обертанні трьох приводних ланок, які пов'язані з шестернями $Gear\ A, Gear\ A_2, Gear\ A_3$, для досягнення бажаної орієнтації рухомої платформи (RyhomaPlatforma). В основі механізму розташовані три приводні ланки. Кожна з цих ланок приводиться в рух окремим сервомотором або іншим типом обертового приводу, пов'язаним із відповідною шестернею ($Gear\ A, Gear\ A_2, Gear\ A_3$). Кожна приводна ланка з'єднана через послідовність шарнірів (Sharnir $B/B_2/B_3$ та Sharnir $C/C_2/C_3$) з рухомою платформою. Шарніри забезпечують обертальні ступені вільності між ланками. Всі з'єднання розташовані таким чином, що їх рухи відбуваються на уявній сфері, що є характерною ознакою СПМ. Коли приводи обертають приводні ланки, це призводить до зміни кутів у шарнірах відповідних кінематичних ланцюгів. Оскільки три такі ланцюги з'єднані з рухомою платформою у різних точках, їхні скоординовані рухи призводять до зміни орієнтації (кутів нахилу, крену та повороту) платформи (RyhomaPlatforma). Ключовою особливістю паралельних механізмів є те, що рухома платформа з'єднана з основою (де розташовані приводи) через кілька незалежних кінематичних ланцюгів. Основа задає геометричні обмеження для платформи, а положення платформи у просторі визначається взаємодією точок C, C_2, C_3 та B, B_2, B_3. Це забезпечує високу жорсткість, точність та швидкість руху, оскільки навантаження розподіляється між кількома ланками. Для досягнення певної орієнтації платформи, система керування (яка не показана на зображенні) повинна обчислювати необхідні кути обертання для кожного з приводів, що вимагає розв'язання оберненої кінематичної задачі СПМ [23, 34].

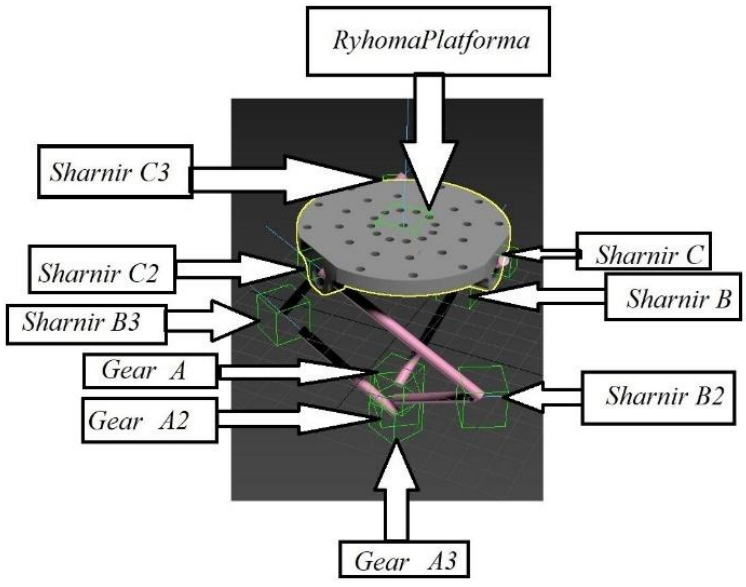
Сферичний механізм має наступні характерні елементи, що наведені нижче:

1. RyhomaPlatforma – центр платформи;
2. Sharnir C, C_2, C_3– шарніри з’єднання платформи та верхніх ланок;
3. Sharnir B, B_2, B_3– шарніри з’єднання верхніх та нижніх ланок;

4. Gear A, A_2, A_3 – шестерні, значення яких отримуються для керування дослідним зразком.



а)



б)

Рис. 2.1 – Структура сферичного механізму з позиціонуючою платформою: а) – виготовлений дослідний зразок на 3д принтері; б) – ієрархічна тривимірна кінематична модель

2.2. Формування геометричної моделі механізму

Розглянуті методи мають суттєві обмеження: метод замкнутого векторного контуру придатний лише для плоских механізмів, а метод перетворення координат потребує складних обчислень та спеціалізованого програмного забезпечення [27]. Використаємо метод, заснований на розрахунку оберненої кінематичної задачі для СПМ, який було модифіковано для спрощення розрахунку та відмови від використання комерційних програмних продуктів [27, 90- 92].

На рис. 2.2 наведено розрахункову схему, в якій робочу систему координат суміщено з площиною $П_o$кола O_k.

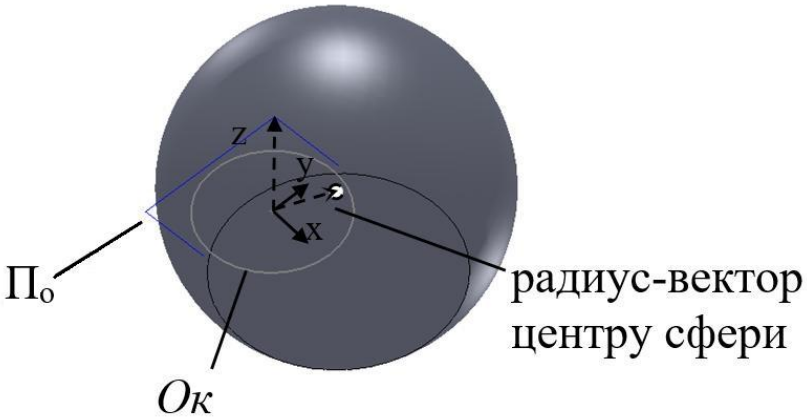


Рис. 2.2 – Геометрична схема розв’язання зворотної задачі кінематики

Коло задане рівнянням:

$$x^2 + y^2 = R_1^2, \tag{2.1}$$

де x та y— координати положення точки у площині $П_o$, R_1— радіус кола.

Сфера задана положенням центру (x_0, y_0, z_0) відносно цієї системи координат та радіусом сфери (R_c). Тоді рівняння сфери має вигляд:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R_c^2, \tag{2.2}$$

де z — координати положення точки в заданій системі координат.

Коло, отримане перетином сфери і По отримано при $z = 0$ та, відповідно, має рівняння:

$$(x - x_0)^2 + (y - y_0)^2 = R_c^2 - {z_0}^2. \quad (2.3)$$

Радіус цього кола (R_2) таким чином має вигляд:

$$R_2 = \sqrt{R_c^2 - {z_0}^2}. \quad (2.4)$$

Тоді точки перетину будуть отримані як розв'язок системи:

$$\begin{cases} x^2 + y^2 = R_1^2 \\ (x - x_0)^2 + (y - y_0)^2 = R_2^2. \end{cases} \quad (2.5)$$

При розв'язані системи рівнянь (2.5) за допомогою математичного середовища Maple отримано наступні корені:

$$B_X = \frac{x_0^4 + R_1^2 \cdot x_0^2 - R_0^2 \cdot x_0^2 + y_0^2 \cdot x_0^2 - \sqrt{F} \cdot y_0}{2 \cdot x_0 \cdot ({x_0}^2 + {y_0}^2)}, \quad (2.6)$$

$$B_Y = \frac{y_0}{2} + \frac{R_1^2 \cdot y_0 - R_0^2 \cdot y_0 + \sqrt{F}}{2 \cdot ({x_0}^2 + {y_0}^2)}, \quad (2.7)$$

$$B_{2X} = \frac{x_{02}^4 + R_1^2 \cdot x_2^2 - R_2^2 \cdot x_2^2 + y_2^2 \cdot x_2^2 - \sqrt{F} \cdot y_2}{2 \cdot x_2 \cdot ({x_2}^2 + {y_2}^2)}, \quad (2.8)$$

$$B_{2Y} = \frac{y_2}{2} + \frac{R_1^2 \cdot y_2 - R_2^2 \cdot y_2 + \sqrt{F}}{2 \cdot ({x_2}^2 + {y_2}^2)}, \quad (2.9)$$

$$B_{3X} = \frac{x_3^4 + R_1^2 \cdot x_3^2 - R_3^2 \cdot x_3^2 + y_3^2 \cdot x_3^2 - \sqrt{F} \cdot y_3}{2 \cdot x_3 \cdot ({x_3}^2 + {y_3}^2)}, \tag{2.10}$$

$$B_{3Y} = \frac{y_0}{2} + \frac{R_1^2 \cdot y_3 - R_3^2 \cdot y_3 + \sqrt{F}}{2 \cdot ({x_3}^2 + {y_3}^2)}, \tag{2.11}$$

де

$$\begin{gathered} F = -{R_1}^4 \cdot {x_0}^2 - {x_0}^6 - {y_0}^4 \cdot {x_0}^2 - 2 \cdot {x_0}^4 \cdot {y_0}^2 + \\ +2 \cdot {x_0}^4 \cdot {R_2}^2 - {x_0}^2 \cdot {R_2}^4 + 2 \cdot {x_0}^4 \cdot {R_1}^2 + \\ +2 \cdot {y_0}^2 \cdot {R_2}^2 \cdot {x_0}^2 + 2 \cdot {R_1}^2 \cdot {y_0}^2 \cdot {x_0}^2 + 2 \cdot {x_0}^2 \cdot {R_1}^2 {R_2}^2. \end{gathered} \tag{2.12}$$

Отримані корені визначають можливі положення точок B, B_2, B_3.

Для визначення положення шарнірів B, B_2, B_3 вздовж осі z розраховуються радіуси R_0, R_2, R_3, які описують максимально можливу відстань проекцій шарнірів від осі z, обумовлену фізичними обмеженнями конструкції механізму. Ці радіуси є статичними параметрами та розраховуються одноразово на основі початкової конфігурації механізму:

$$\begin{cases} R_2 = \sqrt{R_C^2 - z_0^2}, \\ R_3 = \sqrt{R_C^2 - z_2^2}, \\ R_4 = \sqrt{R_C^2 - z_4^2}. \end{cases} \tag{2.13}$$

де R_c – довжина ланки між шарнірами C і B, C_2 і B_2, C_3 і B_3; z_0, z_2, z_3 – положення шарнірів C, C_2, C_3 вздовж осі z.

2.3. Урахування геометричних обмежень та допустимих конфігурацій

Рухома платформа розглядуваного механізму визначається трьома характерними точками C, C_2, C_3, які задають її положення у тривимірному

просторі відносно декартової системи координат (x, y, z). У початковому положенні координати цих точок мають вигляд $C(x_C, y_C, z_C), C_2(x_{C2}, y_{C2}, z_{C2}), C_3(x_{C3}, y_{C3}, z_{C3})$.

Фактично трикутник, утворений цими точками, визначає площину рухомої платформи та дозволяє однозначно відстежувати її орієнтацію у просторі.

Опорні точки основи механізму – B, B_2, B_3 – розташовані у площині XY. Вони утворюють коло з радіусом R та центром у початку координат, що описується співвідношеннями: $B(x_B, y_B, 0), B_2(x_{B2}, y_{B2}, 0), B_3(x_{B3}, y_{B3}, 0)$.

Таким чином, основа задає геометричні обмеження для платформи, а її положення у просторі визначається взаємним розташуванням пар точок C і B,C_2 і B_2 ,C_3 і B_3.

Для визначення допустимих положень точок B, B_2, B_3 необхідно враховувати обмеження, що випливають з геометрії та кінематики механізму. Кожна з систем нелінійних рівнянь потенційно може мати до чотирьох можливих розв'язків. Це зумовлено тим, що точка перетину кола, яка задає геометричне обмеження та сфери з центром у точці C, що задає кінематичне обмеження, може знаходитися в одній з чотирьох комбінацій знаків координат x та y.

Геометричне обмеження описані рівнянням (2.1) – всі точки B лежать на колі з радіусом R_1 та центром у початку координат $O(0, 0)$. Можна представити у вигляді:

$$x_B^2 + y_B^2 = R_1^2, \tag{2.14}$$

де x_B та y_B – координати положення точки B в заданій системі координат.

Кінематичне обмеження представлене у рівнянні (2.2) – відстань між відповідними точками C та B дорівнює довжині ланки механізму R_c, можна описати як:

$$(x_C - x_B)^2 + (y_C - y_B)^2 + {z_0}^2 = R_c^2 \tag{2.15}$$

де x_C та y_C – координати положення точки B в заданій системі координат.

Аналогічні системи рівнянь формуються для пар $C_2 - B_2$ та $C_3 - B_3$.

Тож, загальна кількість можливих комбінацій положень трьох точок B, B_2, B_3 може сягати 64 (4×4×4), що потребує застосування процедури фільтрації для вибору єдиної фізично допустимої конфігурації [27, 29].

Таким чином, задача зводиться до розв'язання трьох систем нелінійних рівнянь, кожна з яких може мати до чотирьох можливих розв'язків, що вимагає розгляду двох додаткових задач:

1. Визначення відносних кутів переміщення точок середньої ланки (B, B_2, B_3), які характеризують зміну положення точок при переході від попереднього стану до прогнозованого, наступного та дають можливість відстежувати не лише абсолютні координати, але й напрямок та величину зміщення кожної точки по колу основи. Таким чином, відносні кути дають можливість за геометричними параметрами механізму досліджувати його реальну динаміку.
2. Обчислення кутів повороту серво двигунів (a, a_2, a_3), що є ключовою задачею, оскільки саме ці кути є безпосередніми керуючими впливами, які подаються на виконавчі пристрої. Правильність їх визначення істотно впливає на точність руху платформи, стабільність роботи системи та її відповідність заданій траєкторії. Від обчислення кутів повороту a, a_2, a_3 залежить можливість коректного перетворення умовних поворотів платформи (α, β, γ) у фізично реалізовані положення серво двигунів.

Через надлишковість отриманих рішень виникає потреба у відборі єдиної фізично допустимої конфігурації механізму, що реалізується у багатокроковому алгоритмі фільтрації.

На першому етапі здійснюється відбір точок, що належать колу з радіусом R_1 (з урахуванням допустимої похибки ε). Для цього покроково аналізуються всі можливі комбінації координат (x, y) точок B, B_2, B_3. Обчислюється евклідова норма відповідного вектора, після чого визначається різниця між цією величиною та радіусом R_1. Якщо модуль різниці не перевищує ε, точка вважається такою, що задовольняє умові належності колу. На цьому етапі для кожної з точок B, B_2, B_3 може існувати кілька допустимих варіантів. З отриманих

варіантів формується множина всіх можливих комбінацій, що надалі підлягають подальшій фільтрації. Результати виконання першого етапу, а саме можливі положення точок, які задовольняють геометричному обмеженню $x_{B_2} + y_{B_2} = R_1^2$ наведено на рис. 2.3.

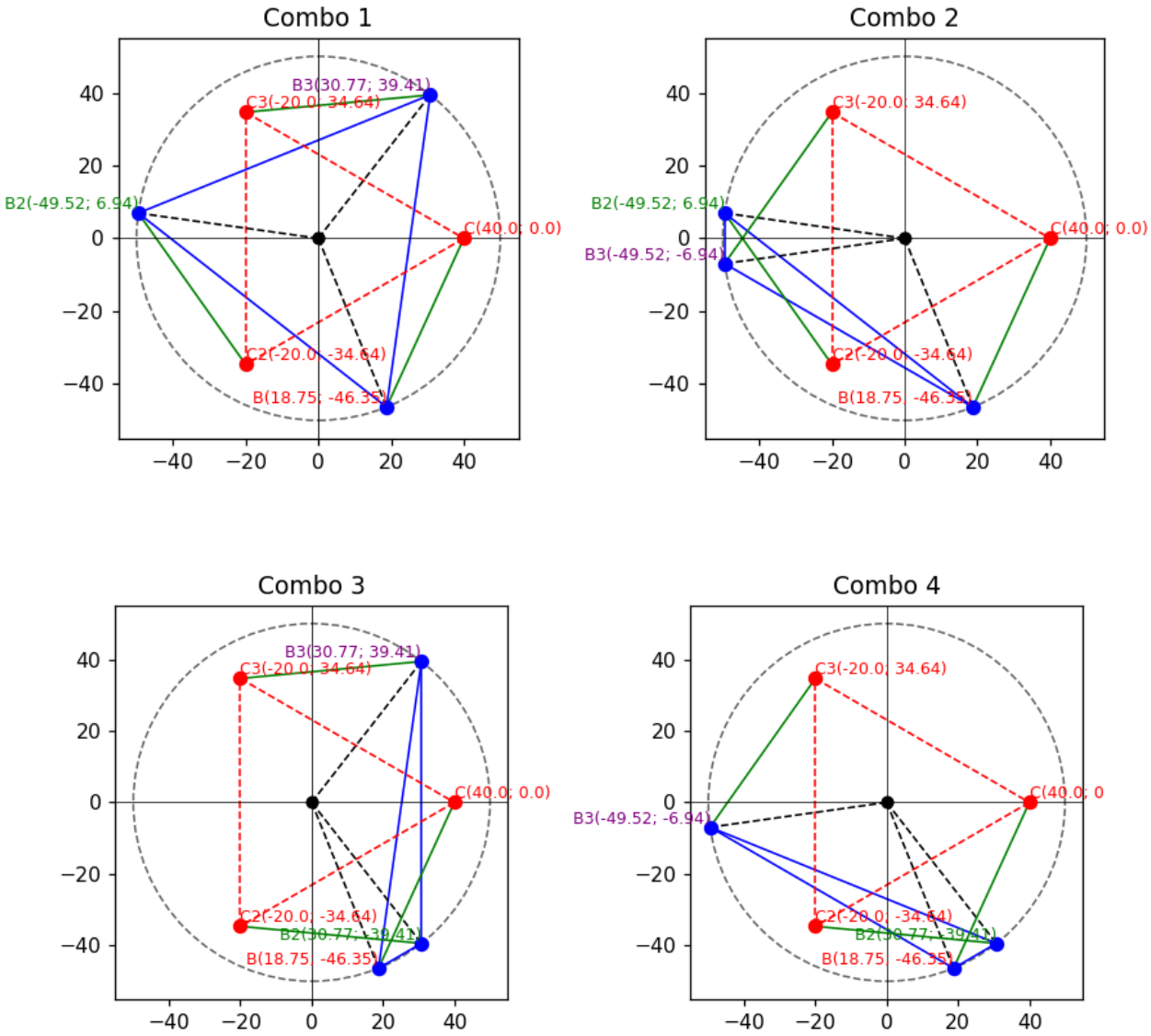
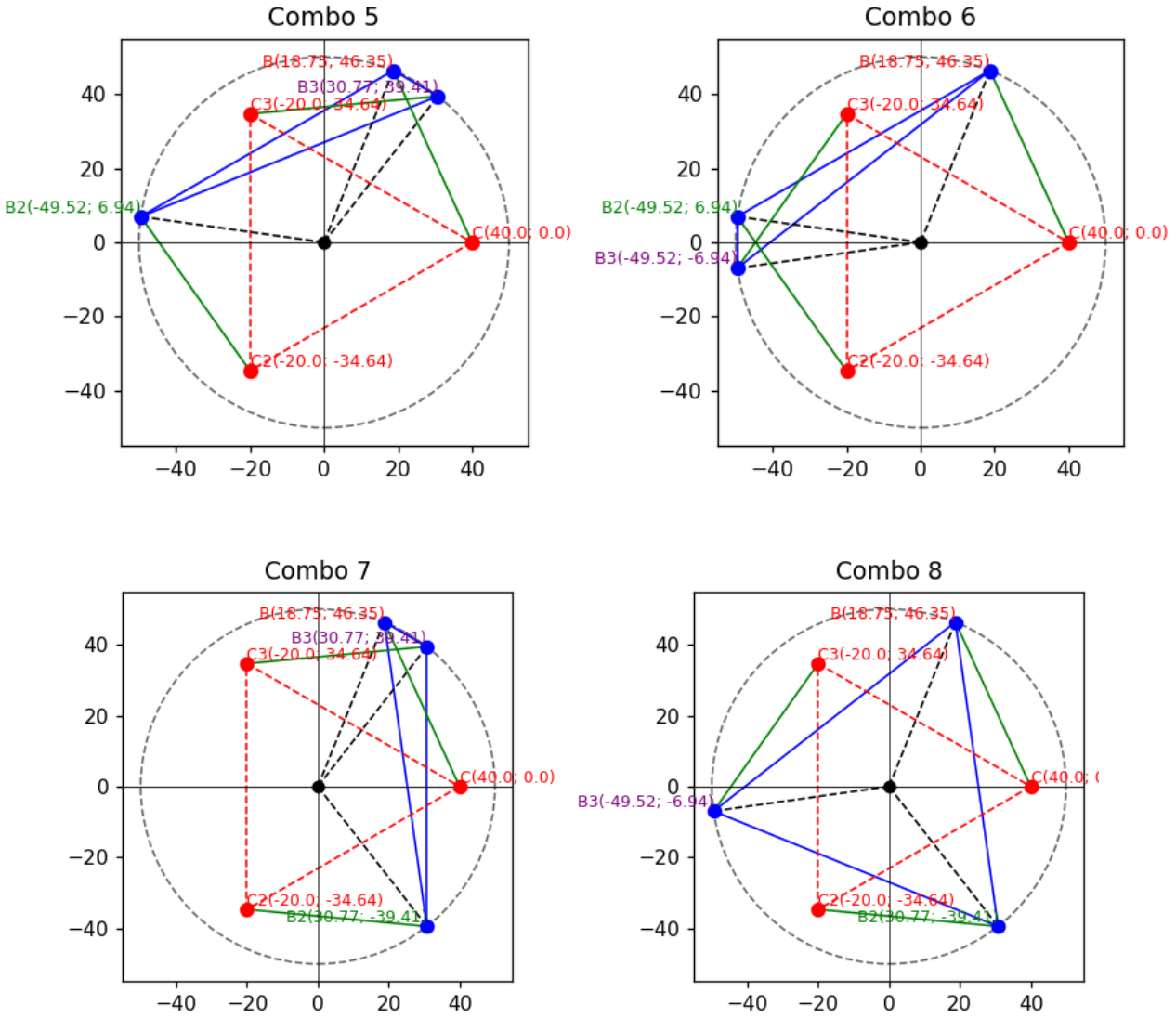


Рис. 2.3 – Визначення множини кандидатів точок B, B_2, B_3 шляхом перевірки належності колу радіуса R_1

Другий етап включає два послідовні кроки перевірки, спрямовані на забезпечення виконання фізичних обмежень механізму: перевірка мінімальної відстані між можливими положеннями точок та перевірка на відсутність перетину ланок.



Продовження Рис. 2.3– Визначення множини кандидатів точок B, B_2, B_3 шляхом перевірки належності колу радіуса R_1

Для уникнення вироджених конфігурацій виконується аналіз попарних відстаней між усіма кандидатами точок B, B_2, B_3. Якщо будь-яка з відстаней виявляється меншою за встановлену граничну величину d_{min} , така комбінація відкидається як некоректна. Ця умова відображає фізичне обмеження механізму, пов’язане з мінімально допустимою шириною з’єднувальної ланки. Результати процедури відсіювання комбінацій, у яких відстань між точками B, B_2, B_3 є меншою за допустиме значення, що відповідає фізичним обмеженням механізму наведено на рис. 2.4.

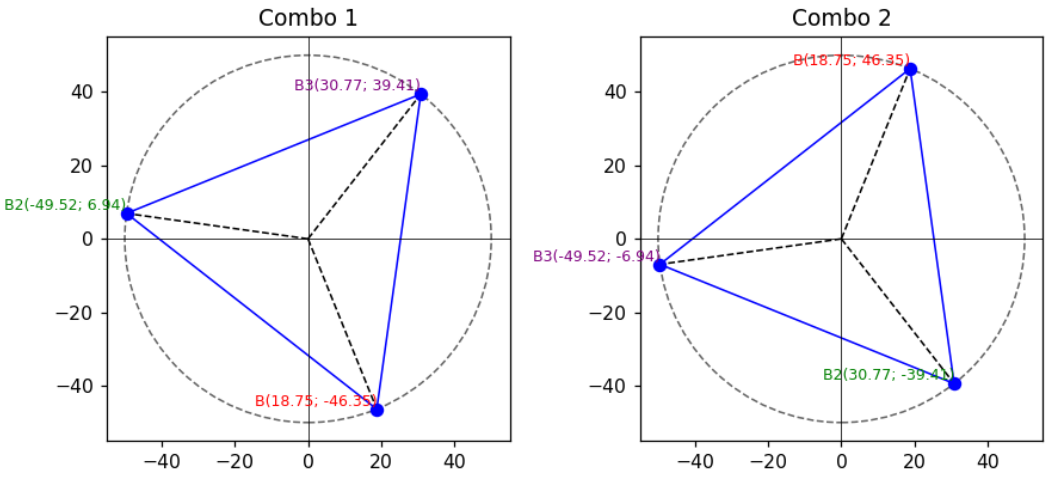


Рис. 2.4 – Результат фільтрації множини кандидатів на основі критерію мінімальної відстані d_{min}

Аналіз орієнтації векторів, що описують з'єднання між точками B і C дозволяє виявити потенційні випадки перетину ланок, що є фізично неможливими у реальній роботі механізму. Всі конфігурації, для яких зафіксовано перетин, відсіюються з подальшого розгляду. Результати цієї перевірки, а саме відсіювання конфігурацій, у яких зафіксовано перетин відрізків B і C, B_2 і C_2, B_3 і C_3, як таких, що є фізично неможливими наведено на рис. 2.5.

Остаточний етап здійснюється за критерієм найбільшої близькості до попереднього положення системи, що дає змогу забезпечує плавність, безперервність та узгодженість руху виконавчих органів. Для цього серед усіх допустимих комбінацій обирається варіант, параметри якого мають мінімальне відхилення від координат і орієнтації системи на попередньому кроці.

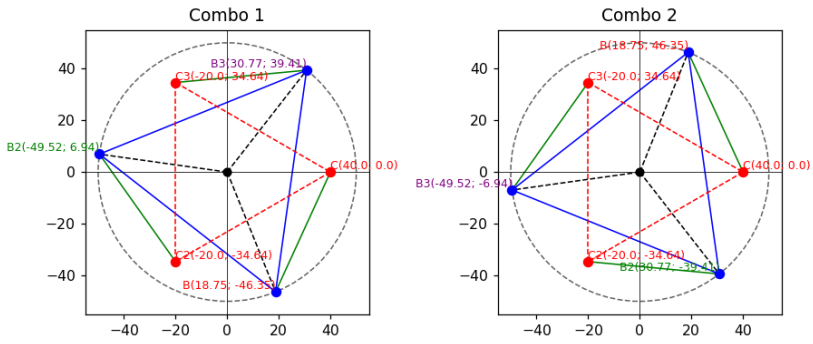


Рис. 2.5 – Результат перевірки на відсутність перетину ланок за допомогою аналізу орієнтації векторів

Такий підхід дозволяє уникнути різких змін положення механізму, стрибкоподібних переміщень та неоднозначностей при виборі конфігурації.

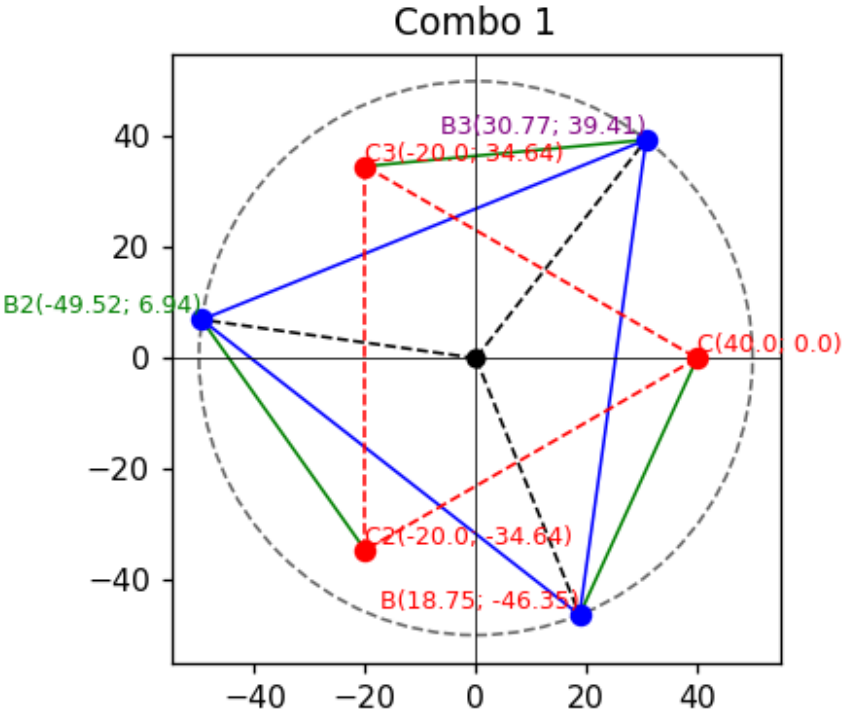


Рис. 2.6 –Остаточний результат відборі єдиної фізично допустимої конфігурації механізму

Послідовне виконання описаних процедур фільтрації та відбору дало можливість поетапно зменшити кількість можливих комбінацій до одного остаточного варіанту (рис. 2.6). Це забезпечує коректне визначення робочої конфігурації механізму та підвищує стабільність керування виконавчими органами системи під час її функціонування.

2.4. Модель орієнтації платформи через матричні перетворення

Поточне положення рухомої платформи відносно початкового визначається трьома кутами повороту навколо координатних осей: α – поворот навколо осі X, β – поворот навколо осі Y, γ – поворот навколо осі Z [9]. Ці кути належать до вхідних параметрів системи і визначають, як саме початкові координати точок C, C_2, C_3 будуть перетворені у нове положення.

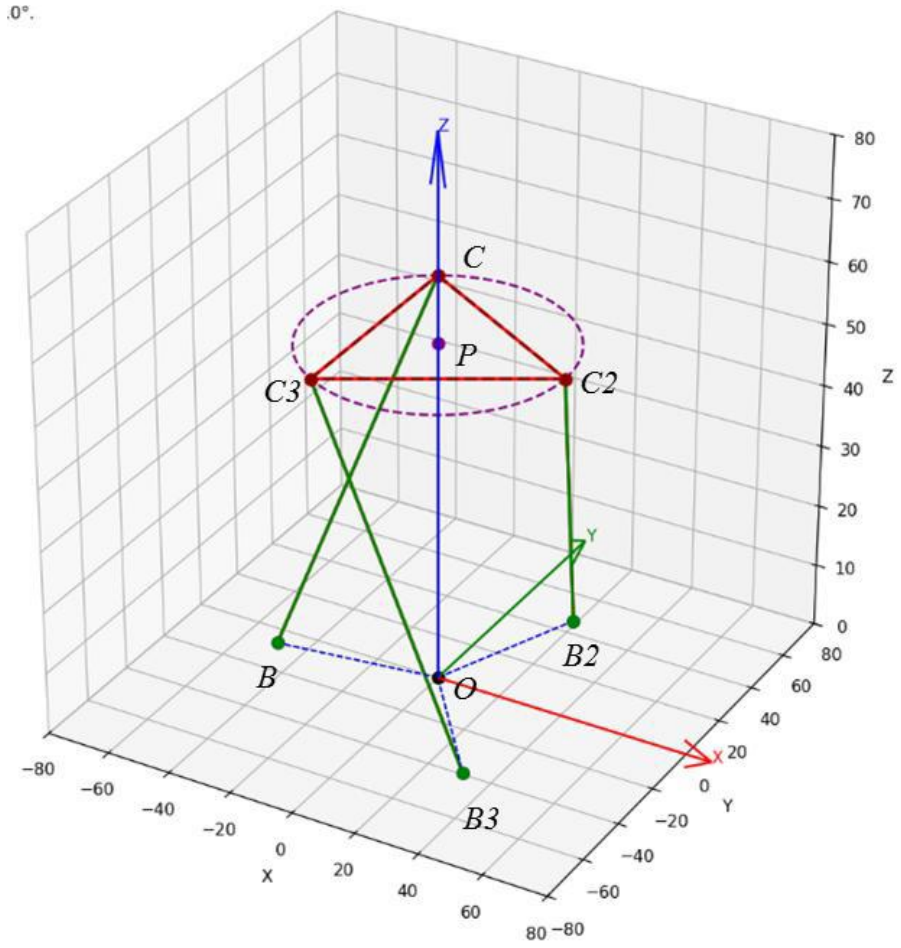


Рис. 2.7 – Схема платформи у початковому положенні

Перехід від початкових координат до поточних здійснюється за допомогою матриць повороту. Для кожної з осей матриці повороту мають вигляд:

– матриця повороту навколо осі X:

$$M_X(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\alpha & -sin\,\alpha \\ 0 & sin\,\alpha & cos\,\alpha \end{pmatrix}, \tag{2.16}$$

– матриця повороту навколо осі Y:

$$M_Y(\beta) = \begin{pmatrix} cos\,\beta & 0 & sin\,\beta \\ 0 & 1 & 0 \\ sin\,\beta & 0 & cos\,\beta \end{pmatrix}, \tag{2.17}$$

– матриця повороту навколо осі Z:

$$M_Z(\gamma) = \begin{pmatrix} cos\,\gamma & -sin\,\gamma & 0 \\ sin\,\gamma & cos\,\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (2.18)$$

Загальна матриця повороту M визначається як послідовний добуток матриць повороту навколо кожної з осей:

$$M = M_X(\alpha) \cdot M_Y(\beta) \cdot M_Z(\gamma), \quad (2.19)$$

Послідовність застосування матриць відповідає обертанню спочатку навколо осі Z, потім навколо осі Y, і нарешті навколо осі X. Такий порядок є стандартним для представлення орієнтації твердого тіла у просторі через кути Ейлера.

Нові координати будь-якої точки платформи C_i^{new} після повороту обчислюються за формулою:

$$C_i^{new} = M \cdot C_i^{old} \quad (2.20)$$

де C_i^{old} – матриця значень останніх попередніх координат відповідних точок платформи, i – номер відповідної точки платформи.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M_X(\alpha) \cdot M_Y(\alpha) \cdot M_Z(\gamma) \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, \quad (2.21)$$

де α, β, γ – кути повороту навколо осей x, y, z, $(x \quad y \quad z)^T$ – матриця значень нових координат відповідних точок платформи, x', y', z' – попередні координати по осі x, y, z, $(x' \quad y' \quad z')^T$ - матриця значень останніх попередніх координат відповідних точок платформи.

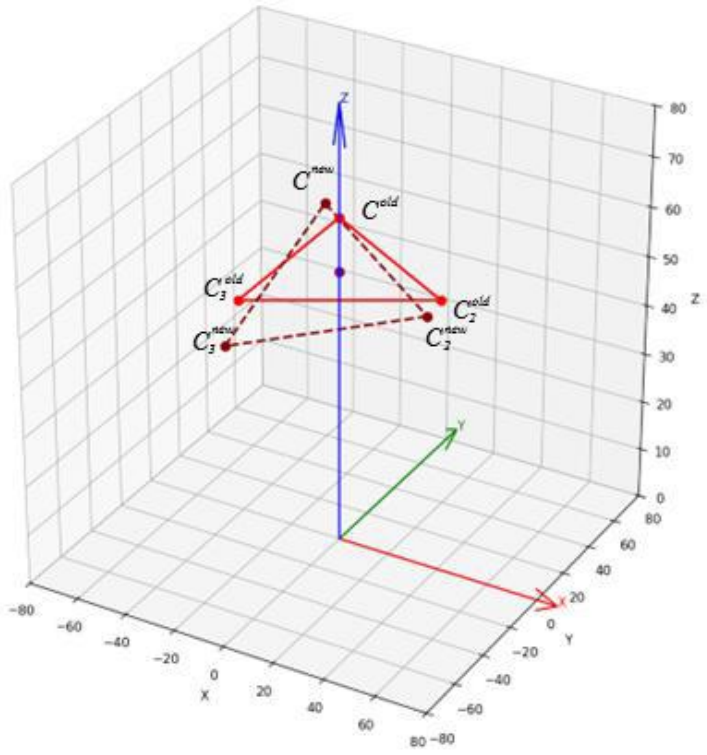


Рис. 2.8 – Результат послідовного застосування матриць M_X, M_Y, M_Z до точок платформи

Важливою особливістю запропонованого підходу є те, що всі наступні повороти здійснюються з уже розрахованими координатами, тобто використовується інкрементальне оновлення: за базові беруться не початкові (нульові) координати, а останні обчислені значення. Це дозволяє уникнути накопичення похибок при послідовних обертаннях та забезпечує плавність руху.

Такий підхід забезпечує коректне врахування орієнтації платформи у просторі та дозволяє однозначно відтворити її положення після довільних обертів. Завдяки використанню апарату матриць обертання задача зводиться до послідовного застосування лінійних перетворень, що зручно як для аналітичних досліджень, так і для програмної реалізації [23, 27]. Крім того, матричний підхід дозволяє легко поєднувати обертання з іншими перетвореннями (масштабування, перенос), що розширює можливості моделювання складних траєкторій руху.

Отримані за допомогою матричних перетворень нові координати точок C', C_2', C_3' використовуються для розрахунку положення точок B, B_2, B_3 шляхом розв'язання систем нелінійних рівнянь, що описують геометричні та кінематичні обмеження механізму.

2.5. Визначення кутового положення на основі інкрементального підходу

Для практичної реалізації керування рухом платформи необхідно визначати не лише абсолютні координати точок B, B_2, B_3, але й відносні кути їх переміщення. Інкрементальний підхід дозволяє відстежувати зміну положення точок при переході від попереднього стану до наступного, що дає можливість оцінювати напрямок та величину зміщення кожної точки по колу основи.

Розглянемо математичну модель визначення координат середніх точок механізму. Робоча платформа описується трьома точками C, C_2, C_3 в системі координат XYZ. Шарніри середньої ланки (точки B, B_2, B_3) обмежені площиною XY, тобто їх координати мають вигляд (x, y), а координата z завжди дорівнює нулю.

Для кожної пари послідовних положень точок B^{last} та B^{new} формується вектор зміщення v, який характеризує переміщення точки по колу. Кут повороту θ визначається на основі співвідношення між довжиною хорди (модулем вектора зміщення) та радіусом кола:

$$\omega = arcsin\frac{|v|}{R_1}, \tag{2.22}$$

де $|v|$ – довжина вектора зміщення точки B, R_1 – радіус кола основи.

Аналогічні обчислення виконуються для точок B_2 та B_3. В результаті отримуються кути a, a_2, a_3, які характеризують зміну положення кожної з точок середньої ланки:

Знайдені значення кутів a, a_2, a_3 є безпосередніми керуючими впливами для серводвигунів, які забезпечують відпрацювання заданого положення платформи. Такий підхід дозволяє перейти від умовних кутів повороту платформи (α, β, γ) до фізично реалізованих кутів повороту серводвигунів [29].

Варто зазначити, що кути повороту a, a_2, a_3 можуть бути як позитивними, так і негативними відносно умовного нуля, що пов'язано з нульовим кутом сервоприводу. Знак кута залежить від того, в якому напрямку (за годинниковою

стрілкою або проти) відбувається зміщення точки відносно попереднього положення.

2.6. Визначення робочої області сферичного паралельного механізму

Робоча область СПМ – це множина просторових положень та орієнтацій рухомої платформи, які можуть бути досягнуті за допомогою допустимих значень кутів сервоприводів. Визначення цієї області є ключовим етапом проєктування та аналізу СПМ, оскільки воно дозволяє оцінити функціональні можливості механізму та виявити недосяжні конфігурації [27, 29].

Процес визначення робочої області складається з послідовних етапів: генерації множини точок-кандидатів у просторі, перетворення просторових координат в орієнтаційні кути, валідації кутів за граничними значеннями, обчислення оберненої кінематики для визначення кутів сервоприводів з перевіркою фізичної реалізованості, та виділення границі робочої області.

На першому етапі створюється регулярна тривимірна сітка точок у декартовому просторі (мм):

$$P = (x, y, z) \in \mathbb{Z}^3 \tag{2.23}$$

де $x \in [-100, 100]$ – діапазон значень по осі абсцис, $y \in [-100, 100]$ – діапазон значень по осі ординат, $z \in [0, 100]$ - діапазон значень по осі аплікат, $\Delta = 5$мм – крок дискретизації, що задає роздільну здатність пошуку.

На другому етапі для кожної точки $p(x, y, z)$ обчислюються три кути Ейлера, що описують орієнтацію платформи, необхідну для досягнення цієї точки:

$$\alpha = arctg\left(\frac{Oy}{Oz}\right), \qquad \beta = arctg\left(\frac{Ox}{Oz}\right), \qquad \gamma = arctg\left(\frac{Oy}{Ox}\right), \tag{2.24}$$

де α – характеризує нахил відносно площини YZ, β – відносно площини XZ, γ – відносно площини XY, Ox, Oy, Oz – проекції на осі X, Y, Z відповідно. Кути подаються в градусах.

Третім етапом є фільтрація кутів за граничними значеннями на відповідність фізичним обмеженням механізму:

$$|\alpha| \leq 46°, |\beta| \leq 46°, |\gamma| \leq 180° \tag{2.25}$$

Точки, для яких хоча б один з кутів виходить за ці межі, виключаються з подальшого розгляду. Обмеження (2.26) для α та β відповідають максимальним кутам нахилу платформи, визначеним конструкцією механізму.

На четвертому етапі для кожної валідної трійки кутів (α, β, γ) розв'язується задача оберненої кінематики, тобто визначаються кути сервоприводів (a, a_2, a_3), які забезпечують задану орієнтацію платформи.

П’ятим етапом є визначення границі робочої області. Після обчислення кутів сервоприводів для всіх точок-кандидатів точки поділяються на дві множини: досяжні та недосяжні (рис. 2.9).

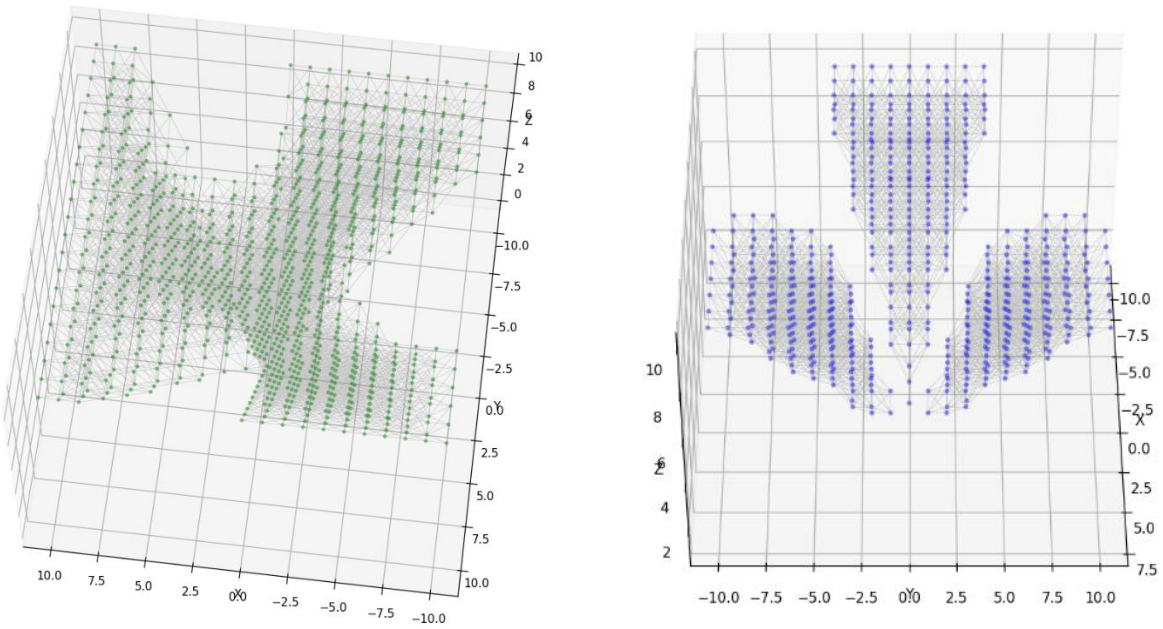


Рис. 2.9 – Множини точок-кандидатів: а) досяжні, б) недосяжні

Точка вважається досяжною, якщо для неї знайдено допустимий розв'язок оберненої кінематики. Для виділення зовнішньої границі робочої області використовується опукла оболонка (рис. 2.11).

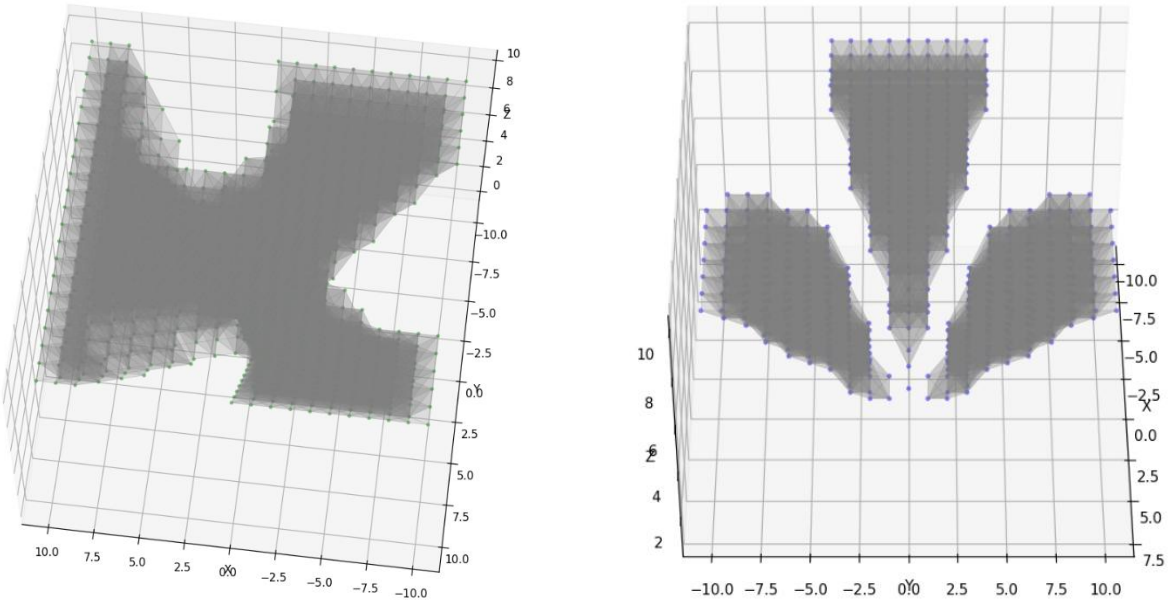


Рис. 2.10 – Візуалізація областей у вигляді опуклої оболонки: а) робоча область; б) неробоча область

Внутрішні точки, які не належать до поверхні опуклої оболонки, виключаються з графічного представлення (рис. 2.11).

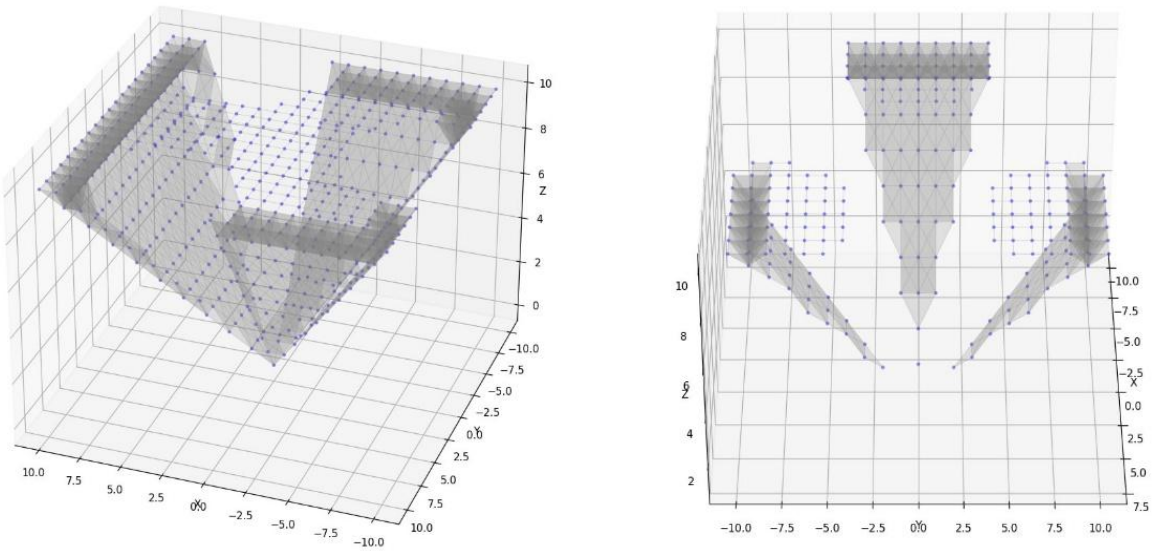
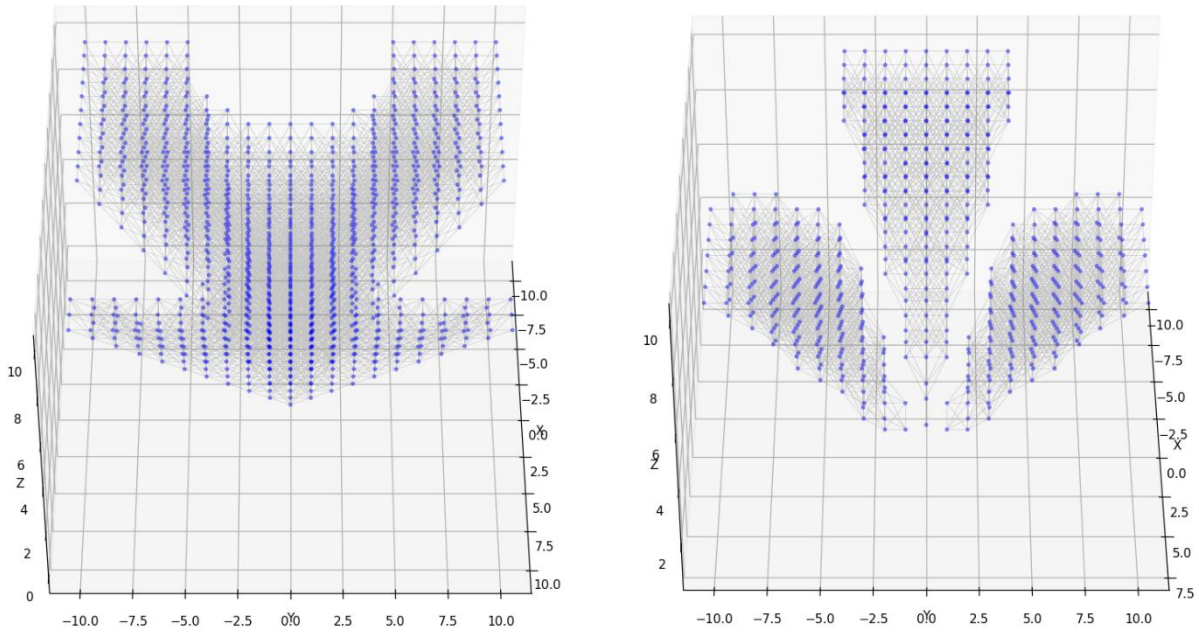


Рис. 2.11 – Візуалізація поверхні опуклої оболонки: а) робоча область; б) неробоча область

Це дозволяє отримати компактний опис робочої області у вигляді набору граничних точок та відповідних їм кутів сервоприводів. Подальша візуалізація виконується у вигляді тривимірної діаграми розсіювання граничних точок (рис. 2.12).



а) б)

Рис. 2.12 – Візуалізація співвіднесення областей: а) робоча, б) неробоча

Результатом є множина досяжних точок P_{valid} та відповідних їм кутів сервоприводів, що визначають робочу область СПМ.

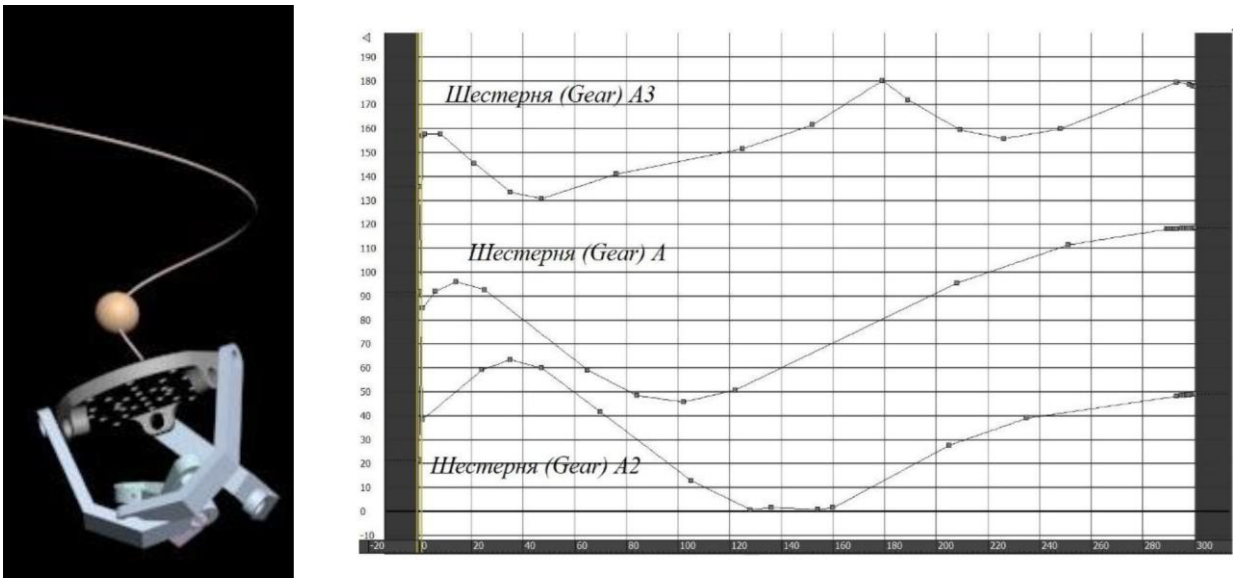
2.7. Розрахунковий експеримент для сферичного паралельного механізму

Для оцінки адекватності математичної моделі проведено серію експериментів із задаванням керуючих впливів α, β та γ у статичних режимах. Реєструвалися сирі дані з інерціального датчика MPU6050 ($gx,\ gy,\ gz$ – кутові швидкості; $ax,\ ay,\ az$ – лінійні прискорення), а також обчислені кути $roll$ та $pitch$. Оскільки значення гіроскопа в усіх тестових точках близькі до нуля ($gx,\ gy,\ gz \approx 0$), це підтверджує, що вимірювання проводилися в усталеному стані після завершення перехідних процесів. Таким чином, при аналізі адекватності

моделі основний акцент зроблено на зіставленні заданих керуючих впливів та виміряних кутів орієнтації.

Отримані результати свідчать, що розроблена математична модель адекватно описує поведінку системи в досліджуваному діапазоні кутів. Високі значення коефіцієнта детермінації, статистична значущість регресійних моделей за F-критерієм Фішера та малі значення похибок апроксимації підтверджують відповідність моделі експериментальним даним і можливість її використання для аналізу та прогнозування параметрів орієнтації рухомої платформи.

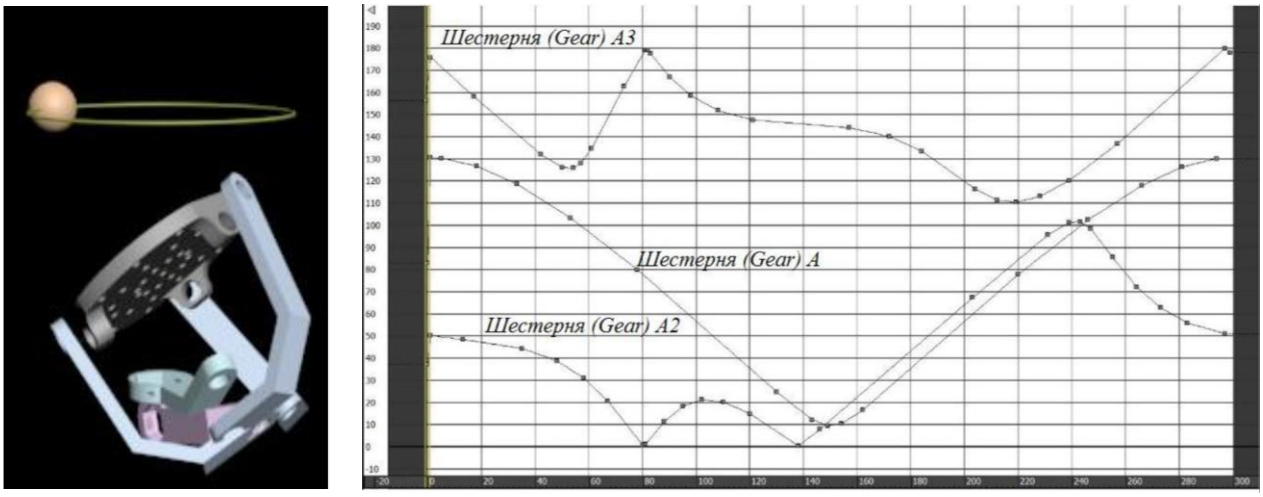
На рис. 2.13 - рис. 2.16 наведені закони переміщення сферичного паралельного механізму, отримані в результаті проведення розрахункового експерименту на основі розробленої математичної моделі механізму. Розрахунок здійснювався шляхом послідовного визначення узагальнених координат та кутів повороту виконавчих ланок для заданих траєкторій руху рухомої платформи. У процесі моделювання враховувалися геометричні параметри механізму, кінематичні обмеження та умови безперервності руху [23, 24. 29].



а) б)

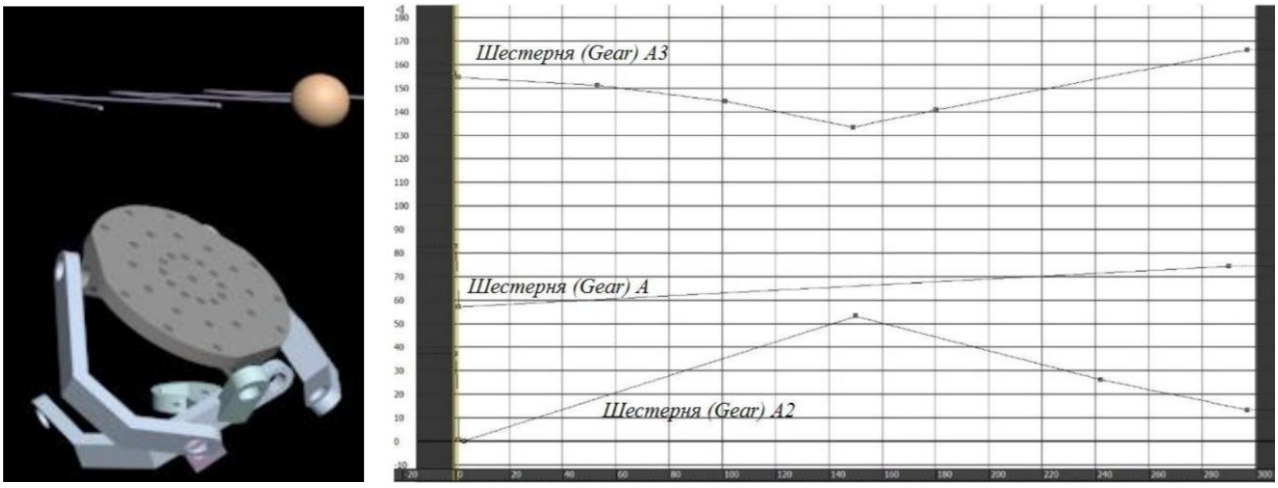
Рис. 2.13 – Моделювання процесу переміщення цілі по траєкторії спіраль: а) скріншот перспективи з ціллю при русі по траєкторії спіраль; б) експериментальні закономірності переміщення

Для оцінки працездатності та особливостей функціонування механізму було обрано декілька типових траєкторій руху: спіраль (рис. 2.13), коло (рис. 2.14), зигзаг (рис. 2.15) та квадрат (рис. 2.16).



а) б)

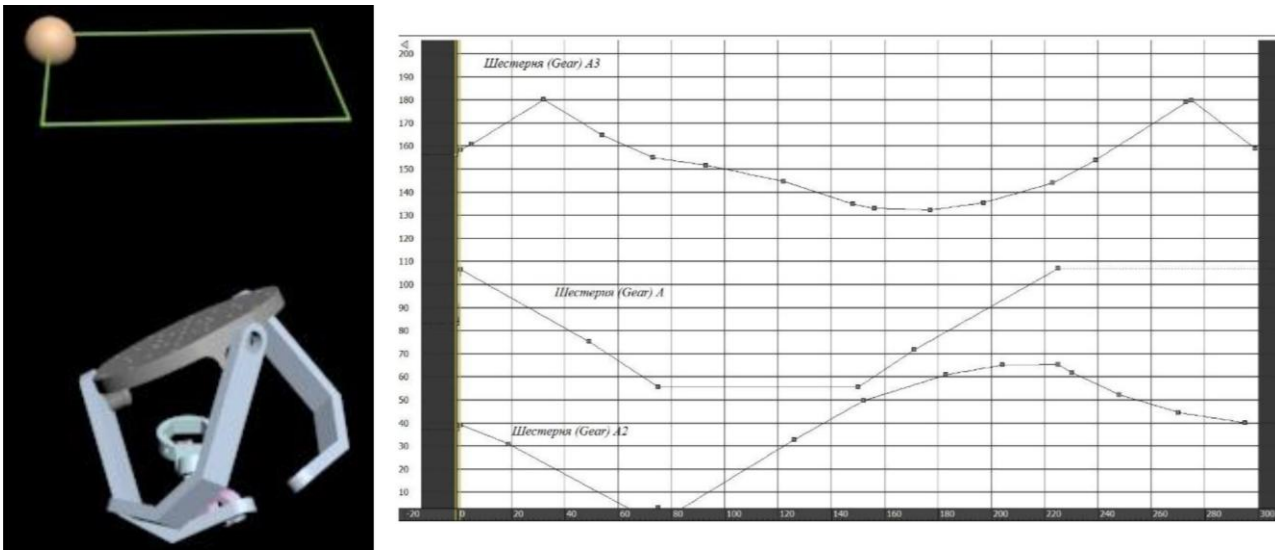
Рис. 2.14 – Моделювання процесу переміщення цілі по траєкторії коло: а) скріншот перспективи з ціллю при русі по траєкторії коло; б) експериментальні закономірності переміщення



а) б)

Рис. 2.15 – Моделювання процесу переміщення цілі по траєкторії зігзаг: а) скріншот перспективи з ціллю при русі по траєкторії зігзаг; б) експериментальні закономірності переміщення

Вибір саме таких траєкторій обумовлений необхідністю перевірки роботи механізму при різних режимах зміни положення та орієнтації рухомої платформи, зокрема при плавному безперервному русі, періодичних змінах напрямку та наявності ділянок із різкими переходами між окремими положеннями.



а) б)

Рис. 2.16 – Моделювання процесу переміщення цілі по траєкторії квадрат: а) скріншот перспективи з ціллю при русі по траєкторії квадрат; б) експериментальні закономірності переміщення

Отримані закони переміщення дозволяють оцінити характер зміни координат та кутових параметрів механізму в часі, а також проаналізувати узгодженість руху виконавчих ланок під час відпрацювання заданих траєкторій. Результати розрахункового експерименту підтверджують коректність побудованої математичної моделі та можливість її використання для подальшого дослідження кінематики сферичного паралельного механізму, аналізу його робочої області та розробки алгоритмів керування.

2.8. Висновки до розділу 2

Розроблено математичну модель сферичного паралельного механізму, що охоплює всі ключові аспекти, необхідні для його керування: геометричний опис механізму, модель орієнтації платформи на основі матричних перетворень, інкрементальний підхід до визначення кутового положення, врахування геометричних та кінематичних обмежень, багатокроковий алгоритм фільтрації для вибору єдиної допустимої конфігурації, визначення робочої області, розрахунковий експеримент.

Геометрична модель задає початкові співвідношення між елементами механізму та визначає робочий простір. В основу геометричної моделі покладено розв'язання зворотної задачі кінематики з використанням законів аналітичної геометрії, що дозволяє відмовитися від складних матричних перетворень систем координат кожної ланки відносно іншої.

Модель орієнтації через матриці повороту забезпечила коректне перетворення просторового положення платформи у координати точок C, C_2, C_3. Матричний підхід дозволив однозначно описати довільну орієнтацію платформи у просторі за допомогою трьох кутів (α, β, γ) та забезпечує зручну програмну реалізацію.

Інкрементальний підхід дозволив перейти від умовних поворотів платформи до безпосередніх керуючих впливів для серводвигунів. Шляхом обчислення кутів a, a_2, a_3 на основі векторів зміщення точок B, B_2, B_3 забезпечується зв'язок між просторовою орієнтацією платформи та фізичними кутами повороту приводів.

Врахування геометричних та кінематичних обмежень дало можливість гарантувати, що отримані конфігурації є фізично реалізованими. Системи нелінійних рівнянь, що описують ці обмеження, можуть мати множину розв'язків, тому алгоритм фільтрації забезпечує вибір єдиної коректної комбінації положень ланок.

Особливе значення має можливість одночасного відстеження як умовних кутів повороту платформи (α, β, γ), так і абсолютних кутів серводвигунів

(a, a_2, a_3). Це поєднання є критично важливим для систем керування, де необхідно враховувати не лише локальні відносні зміщення, але й глобальне положення механізму у просторі.

Запропоновано методику визначення робочої області сферичного паралельного механізму, що ґрунтується на послідовному аналізі просторових положень платформи, перевірці їх відповідності кінематичним та конструктивним обмеженням і розв’язанні задачі оберненої кінематики. У результаті отримано множину досяжних конфігурацій механізму та побудовано границю його робочої області, що дозволяє оцінити функціональні можливості СПМ і визначити допустимі режими його експлуатації.

Розрахунковий експеримент та експериментальна перевірка підтвердили адекватність розробленої математичної моделі. За результатами статистичної оцінки отримано коефіцієнти детермінації $R^2 = 0{,}9996$ для кута $roll$ та $R^2 = 0{,}9990$ для кута $pitch$, середньоквадратична похибка становить 0,238° та 0,341° відповідно, а максимальна відносна похибка не перевищує 2,11%.

Отримані результати свідчать про коректність моделювання орієнтації рухомої платформи в межах досліджуваного діапазону кутів. Розроблена математична модель забезпечує визначення просторового положення платформи та формування керуючих сигналів для серводвигунів, що підтверджує її придатність для використання в робототехнічних системах керування, поєднуючи математичний апарат із практичними методами комп'ютерного моделювання.

РОЗДІЛ 3 АЛГОРИТМІЧНА ТА ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ

Забезпечення високої точності керування сферичними паралельними механізмами [3, 23] є комплексним завданням, яке вимагає врахування багатьох факторів. Основними серед них є використання високоточних датчиків для вимірювання положення і швидкості виконавчих органів та застосування ефективних методів ідентифікації параметрів руху елементів СПМ та алгоритмів керування, що компенсують похибки механізму. Тож актуальною задачею дослідження є розробка методу ідентифікації параметрів руху елементів сферичного паралельного механізму.

Для досягнення мети пропонується застосувати алгоритм комплементарного фільтру, адаптований для впровадження в системи керування СПМ [30]. Впровадження методу забезпечить підвищення точності оцінювання стану системи при наявності шумів в реальних промислових умовах.

Окрім цього, важливим етапом дослідження є програмна реалізація системи керування сферичним паралельним механізмом, що включає розробку програмних модулів збору та обробки даних з датчиків, реалізацію алгоритмів фільтрації та ідентифікації параметрів руху, а також формування керуючих впливів для виконавчих органів механізму. Це дозволяє забезпечити узгоджену роботу всіх компонентів системи та створює основу для практичного впровадження розроблених методів у реальних системах керування СПМ

3.1. Алгоритм комплементарного фільтра для ідентифікації параметрів руху

Останні дослідження демонструють значний прогрес у використанні фільтрів Махоні [15], Маджвіка, Калмана [16, 17] для ідентифікації параметрів руху елементів сферичного паралельного механізму. Для боротьби зі змінними характеристиками шумів та динаміки СПМ досліджується застосування

адаптивних фільтрів [18], які здатні в реальному часі оцінювати та коригувати параметри шуму, що забезпечує кращу якість ідентифікації параметрів руху, особливо при невідомих або змінних зовнішніх впливах. Проте, застосування складних фільтрів, особливо адаптивних, вимагає значних обчислювальних ресурсів, що є обмеженням для систем керування в реальному часі з високою частотою оновлення. Значні труднощі для точної ідентифікації параметрів руху, особливо в широкому діапазоні робочого простору викликає і складна нелінійна кінематика СПМ і залежність точності ідентифікації параметрів руху безпосередньо від якості та точності використовуваних сенсорів.

Для СПМ комплементарний фільтр [30] має певні переваги, адже є значно простішим в реалізації та потребує менше обчислювальних ресурсів порівняно зі складнішими фільтрами, що може бути важливим для систем керування СПМ в реальному часі з обмеженими апаратними ресурсами. Комплементарні фільтри зазвичай мають менше параметрів для налаштування, що спрощує процес їх застосування та можуть ефективно поєднувати дані з гіроскопа та акселерометра.

Проведений аналіз останніх досліджень [14] доводить, що використання комплементарного фільтра може бути корисним для оцінки орієнтації кінцевого положення рухомої платформи СПМ, особливо в застосуваннях, де важлива обчислювальна ефективність та простота реалізації.

Важливу роль в структурі системи керування сферичними паралельними механізмами відіграють датчики, що забезпечують отримання точної інформації про орієнтацію та прискорення механізму в просторі. Ця інформація є необхідною для реалізації різних функцій керування, таких як: забезпечення стійкого положення рухомої платформи СПМ (навіть при наявності зовнішніх впливів); визначення точного положення рухомої платформи в просторі для виконання заданих маніпуляцій; забезпечення руху платформи за заданою траєкторією; компенсація впливу вібрацій, ударів та інших зовнішніх факторів на роботу механізму.

Використання універсального мікроелектромеханічного системного модуля (MEMS) для відстеження руху в системі керування СПМ дозволяє створювати високоточні та надійні роботизовані системи. Прикладом такого

модуля є MPU6050 – високоінтегрований 6-осьовий пристрій для відстеження руху, який поєднує в собі 3-осьовий гіроскоп, 3-осьовий акселерометр [57, 58, 93]. Гіроскоп вимірює кутову швидкість обертання СПМ навколо трьох осей, акселерометр – лінійне прискорення вздовж цих осей.

У разі використання тільки гіроскопа кут нахилу платформи обчислюється за допомогою дискретного інтегрування швидкості обертання. Однак у MEMS гіроскопа є недолік, який називається дрейфом нуля. При зупинці обертання гіроскопа він все ще показуватиме значення відмінне від нуля. Крім того недоліком такого рішення є застосування процедури дискретного інтегрування, яка дає неточний результат вимірювань та сприяє поступовому накопиченню помилки обчислення кута через обмежену точність змінних мікроконтролера. За допомогою акселерометра також можна обчислити кути нахилу платформи, проте будь-який зовнішній вплив на датчик вноситиме помилку в обчислення кута нахилу. Частково зняти вплив зовнішніх сил можна за допомогою ФНЧ, але це приведе до значного зниження швидкодії. Тому для усунення недоліків використання гіроскопа і акселерометра доцільно об’єднати показання цих двох приладів.

Для оцінки орієнтації механізму в просторі отримані показання датчиків пропонується фільтрувати для видалення шуму за допомогою комплементарного фільтра. Задача фільтрації – максимально наблизити виміряне значення величини до дійсного, а отже, мінімізувати величину шуму вимірювань. Вплив шуму враховується наступним чином:

$$z[k] = x[k] + \xi[k], \tag{3.1}$$

де $z[k]$ — виміряне значення на k кроці, $x[k]$ — дійсне значення на k кроці, $\xi[k]$ — величина шуму вимірювань на k кроці.

Принцип роботи комплементарного фільтра заснований на тому, що на кожному поточному k кроці отримується виміряне значення, на основі якого розраховується прогноз на наступне $k + 1$ вимірювання. Після отримання нового $k + 1$ вимірювання визначається відфільтроване значення $x[k]$ виходячи із

виміряного значення та прогнозованого. Ступінь довіри прогнозованим значенням і результатам вимірювання виражається у вагових коефіцієнтах фільтра δ та ρ. Фільтрація дозволяє отримати дійсне значення на даному кроці $x[k]$, швидкість його зміни $\dot{x}[k]$ та прогноз $x^p[k-1]$, які можуть використовуватись в системі керування для розрахунку необхідних команд для сервоприводів СПМ, що забезпечують досягнення бажаного положення рухомої платформи.

Для роботи фільтра потрібно визначити наступні параметри:

1. Оновлення фільтра:

$$x[k] = x^p[k-1] + \delta(z[k] - x^p[k-1]), \tag{3.2}$$

де $x^p[k-1]$ – прогнозоване значення на крок k, отримане на $k-1$ кроці; $\delta \in [0.9; 0.98]$ – коефіцієнт довіри.

Значення коефіцієнта δ в діапазоні $0.9 - 0.98$ визначає ступінь довіри до даних акселерометра. Чим вище значення δ, тим більша довіра до акселерометра та сильніше фільтруються високочастотні шуми гіроскопа. При нижчих значеннях δ більше враховуються дані гіроскопа, що забезпечує краще відстеження динамічних рухів.

2. Оцінка швидкості зміни параметра:

$$\dot{x}[k] = \dot{x}^p[k-1] + \frac{\rho}{T}(z[k] - x^p[k-1]), \tag{3.3}$$

де $\dot{x}[k]$ – відфільтроване значення швидкості зміни параметру на кроці k ; $\dot{x}^p[k-1]$ – прогнозоване значення швидкості зміни параметру на крок k, отримане на $k-1$ кроці; T – інтервал дискретизації; ρ – коефіцієнт для швидкісної складової.

Для роботи фільтра потрібно визначити наступні параметри:

$$x^p[k-1] = x[k-1] + T\dot{x}[k], \tag{3.4}$$

$$\dot{x}^p[k-1] = \dot{x}[k-1].$$

Алгоритм комплементарного фільтра може використовуватись не тільки для процедури фільтрації, а також для поєднання показань гіроскопу і акселерометра.

Дані, зчитані з акселерометра та гіроскопа, надходять у вигляді цифрових значень, що безпосередньо не відповідають фізичним величинам, таким як прискорення [м^2/с] або кутова швидкість [º/с]. Щоб отримати дані в стандартних одиницях вимірювання, необхідно здійснити попереднє перетворення за допомогою процесу масштабування з врахуванням відповідних коефіцієнтів чутливості датчика і його характеристик.

Позначимо масштабовані показання гіроскопу, виміряні на k кроці, як $z_g[k]$, а акселерометра – $z_a[k]$. Тоді прогнозоване значення кута нахилу платформи на крок k, отримане на $k-1$ кроці може бути обчислене як:

$$x^p[k-1] = x[k-1] + T * z_g[k], \tag{3.5}$$

Замінивши відповідні параметри в рівнянні фільтра (3.2), з урахуванням прогнозу (3.5) отримаємо алгоритм розрахунку кута нахилу платформи за допомогою комплементарного фільтра:

$$x[k] = (1-\delta)\big(x[k-1] + T * z_g[k]\big) + \delta * z_a[k]. \tag{3.6}$$

Згідно рівнянню (3.6) підсумкова величина кута нахилу є сумою інтегрованого значення гіроскопа і миттєвого значення акселерометра. На кожному етапі інтегрування коригується інтеграл кута нахилу за допомогою показань акселерометра. Ступінь корекції визначається коефіцієнтом δ, вибір якого залежить від величини дрейфу нуля гіроскопа, від швидкості накопичення

помилок обчислення та умов використання СПМ. Описаний алгоритм пояснюється схемою обробки інформації, представленою на рис. 3.1.

Для забезпечення точної оцінки орієнтації та прискорення необхідна висока частота збору даних з гіроскопа та акселерометра. Перед використанням сенсори необхідно відкалібрувати для компенсації систематичних похибок. Температура може впливати на характеристики сенсорів, тому необхідно передбачити компенсацію температурних змін.

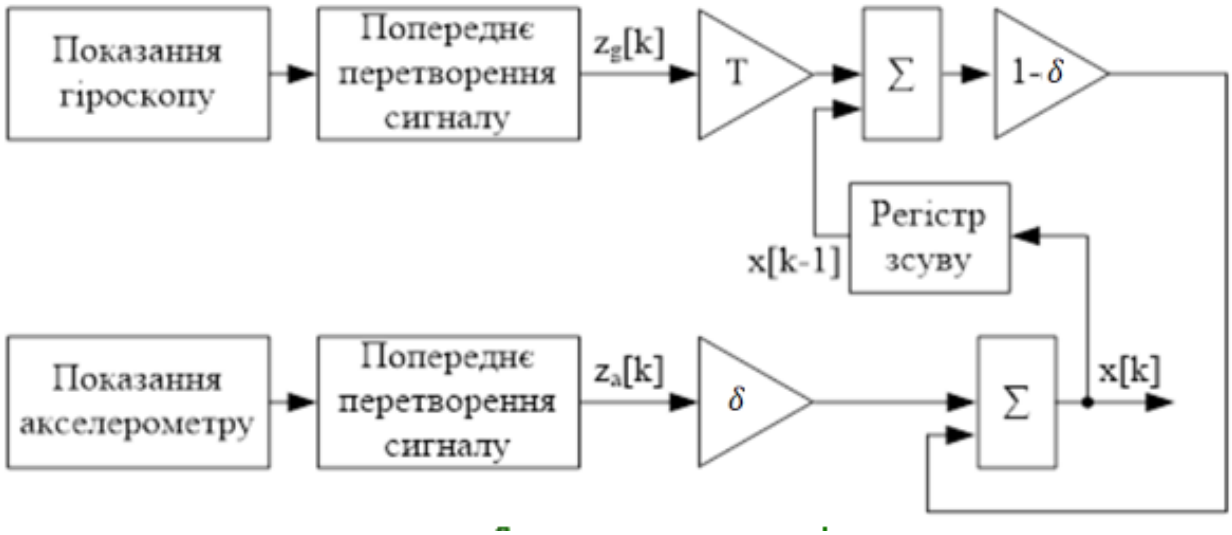
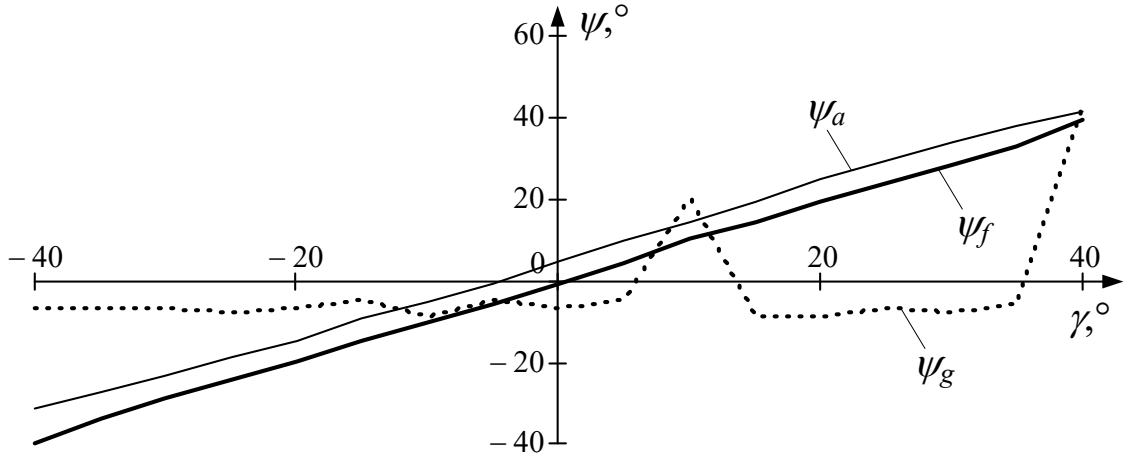


Рис. 3.1 – Блок-схема алгоритму комплементарного фільтра

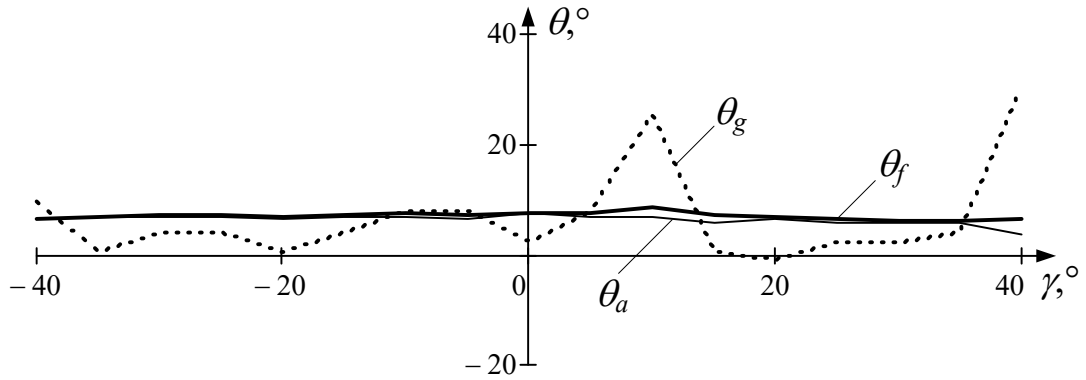
У ході експерименту було здійснено поетапну інтеграцію апаратних та програмних компонентів системи з метою дослідження просторового переміщення платформи СПМ та валідації її фактичного положення за допомогою інерціального модуля. Результати досліджень наведені на рис. 3.2.

До персонального комп’ютера було підключено два мікроконтролери: Arduino Uno [59, 60], що забезпечує керування положенням платформи шляхом подачі сигналів на серводвигуни та Arduino Pro Micro [59], призначений для зчитування даних з інерціального модуля MPU6050, розміщеного безпосередньо на рухомій частині СПМ. Після налаштування серійного зв’язку з обома пристроями було запущено програмне забезпечення, яке послідовно надсилало команди Arduino Uno для встановлення платформи у визначене кутове

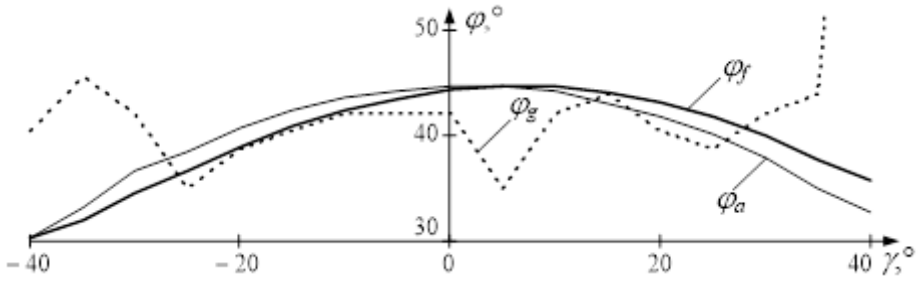
положення γ вздовж осі OY, починаючи з -40° та поступово змінюючи його з кроком 5° до +40° включно.



а) результати вимірювань кута ψ (акселерометром ψ_a і гіроскопом ψ_g) та застосування фільтра ψ_f



б) результати вимірювань кута θ (акселерометром θ_a і гіроскопом θ_g) та застосування фільтра θ_f



с) результати вимірювань кута ϕ (акселерометром ϕ_a і гіроскопом ϕ_g) та застосування фільтра ϕ_f

Рис. 3.2. – Результати експериментальних досліджень

На кожному з етапів, після стабілізації механізму в заданій позиції, на Arduino Pro Micro надсилався запит на зчитування просторових даних з MPU6050. Таким чином, було забезпечено відповідність між теоретично встановленим кутом і фактично зафіксованими показниками модуля, що дозволяє надалі здійснити порівняльний аналіз і оцінити точність керування рухомою платформою. Експеримент відбувався в контрольованому середовищі з урахуванням мінімізації зовнішніх вібрацій та нестабільності живлення, що є критично важливим для коректного зчитування інерціальних даних. Під час руху СПМ, комплементарний фільтр забезпечив плавне та точне відстеження змін орієнтації рухомої платформи, динамічна похибка (відставання або перевищення фактичного кута) знаходиться в межах 0,8º. Відносна простота структури комплементарного фільтра полегшує його налаштування та реалізацію на різних апаратних платформах, що робить його зручним для використання в системах керування СПМ для верифікації алгоритмів позиціонування та відтворення траєкторій.

3.2. Загальна структура програмного забезпечення

На основі математичної моделі СПМ, розробленої в розділі 2, виконано програмну реалізацію системи керування. Розроблене програмне забезпечення реалізує повний цикл обробки даних: від отримання кутів орієнтації платформи до формування керуючих сигналів для серводвигунів.

В залежності від конкретного завдання та експлуатаційних вимог можуть застосовуватися різні методи керування СПМ: керування на основі приводів (електричних, гідравлічних або пневматичних), керування на основі сенсорів (енкодери, акселерометри, датчики сили), керування на основі оцінювання просторової орієнтації голови людини або іншого об’єкта в системі комп’ютерного зору, замкнене керування зі зворотним зв'язком та адаптивне керування [20, 94].

Кожен з цих методів має свої переваги: електричні приводи забезпечують точне позиціювання, гідравлічні та пневматичні — високе зусилля; сенсори

надають зворотний зв'язок у реальному часі; комп’ютерний зір забезпечує безконтактну, інтуїтивну взаємодію та керування в реальному часі; замкнені системи безперервно коригують керуючі впливи на основі похибки; адаптивні стратегії підлаштовуються до змінних зовнішніх умов [23]. В даній роботі основна увага приділяється комбінації сенсорного керування (акселерометр та гіроскоп) з інтелектуальною обробкою даних.

Особливу увагу в межах розглянутого підходу приділено керуванню на основі оцінювання просторової орієнтації голови людини або іншого об’єкта в системі комп’ютерного зору [72, 73]. У цьому випадку вхідні дані формуються безпосередньо з відеопотоку, а просторові координати ключових точок об’єкта використовуються для обчислення кутів орієнтації, що інтерпретуються як керуючі параметри СПМ. Такий підхід дозволяє реалізувати інтуїтивне безконтактне керування, у якому рухи користувача безпосередньо відображаються у зміні положення робочої платформи механізму.

Застосування методів комп’ютерного зору для оцінювання просторової орієнтації об’єкта забезпечує високу швидкодію формування керуючих сигналів та адаптацію системи до динамічних змін у сцені. Використання алгоритмів виділення та трекінгу характерних точок дозволяє підвищити стійкість системи до шумів та часткових перекриттів, а також забезпечує стабільну роботу в режимі реального часу без необхідності використання контактних датчиків або додаткових механічних інтерфейсів.

Програмне забезпечення системи керування СПМ складається з кількох взаємопов'язаних модулів, кожен з яких виконує окрему функцію в загальному циклі керування. До складу програмного комплексу входять: модуль визначення положення платформи на основі розв'язання оберненої задачі кінематики, модуль обробки відеоданих для відстеження об'єкта керування в реальному часі, модуль фільтрації сигналів сенсорів для ідентифікації параметрів руху.

Структурна схема програмного забезпечення наведена на рис. 3.3.

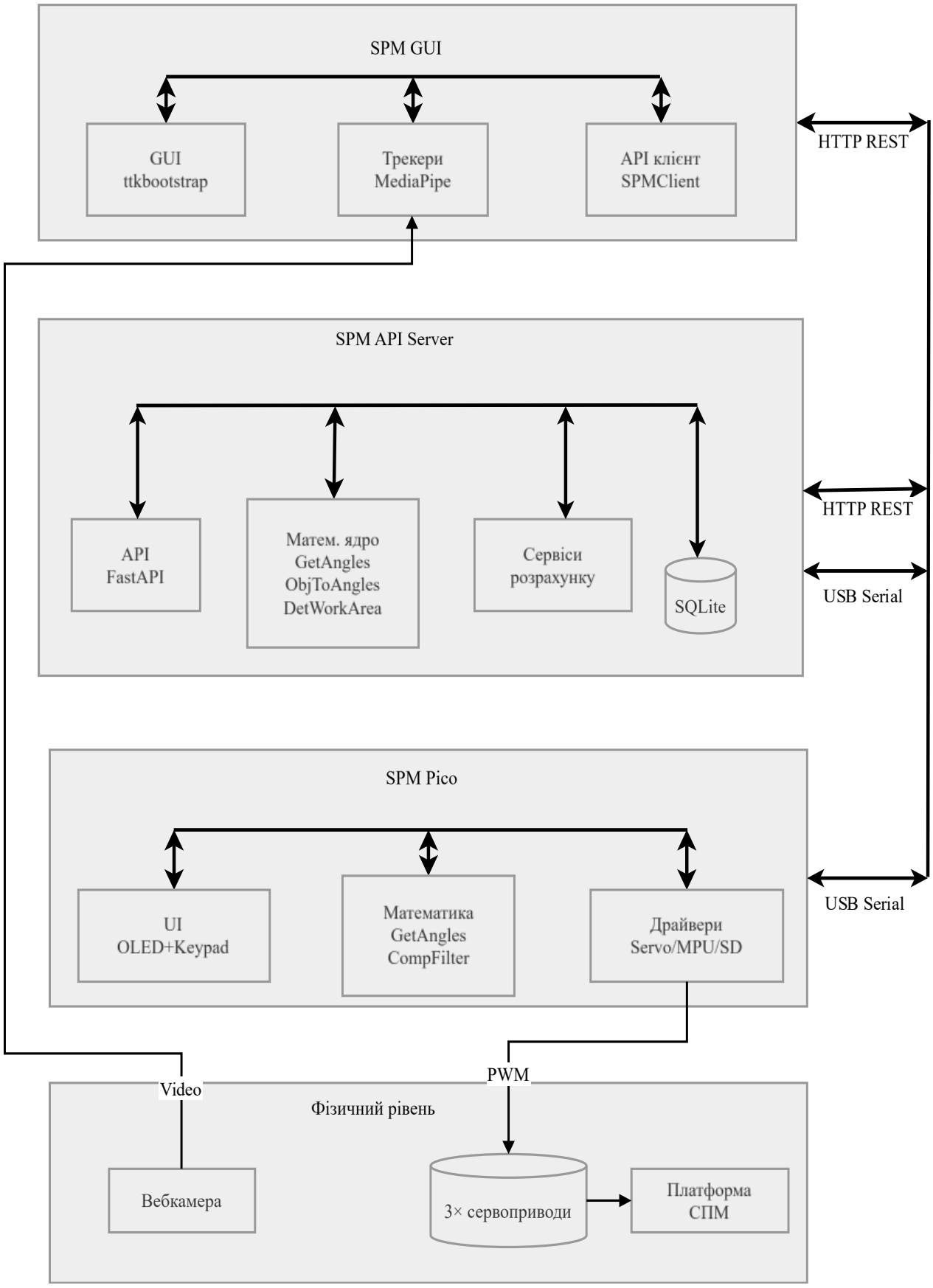


Рис. 3.3 – Структурна схема програмного забезпечення

3.3. Програмна реалізація алгоритму визначення положення платформи

Програмна реалізація алгоритму визначення положення платформи побудована на мові Python [66], що забезпечує гнучкість розробки та інтеграції з різноманітними апаратними засобами. Використовується такі бібліотеки [22, 67, 72, 73, 95-98]:

1. ttkbootstrap (Tkinter) – для створення графічного інтерфейсу користувача на основі Tkinter з розширеними стилями та сучасними елементами керування, що забезпечує зручну реалізацію десктопних GUI-додатків.
2. FastAPI – високопродуктивний веб-фреймворк для Python, призначений для побудови REST API, що забезпечує швидку обробку запитів, асинхронну взаємодію та автоматичну генерацію документації.
3. Numpy, Math – стандартні бібліотеки Python для розширених математичних обчислень.
4. OpenCV – для обробки зображень, що забезпечує кадрування, зміну розміру, обертання зображень, перетворення кольорових зображень у чорно-білі, розмиття, розпізнавання облич;
5. MediaPipe – для обробки потокового відео та задач комп’ютерного зору, що забезпечує детекцію, трекінг і відстеження ключових точок об’єктів (руки, обличчя, поза) у реальному часі з подальшим використанням отриманих координат для аналізу та керування.
6. Matplotlib – для побудови графіків та візуалізації даних у Python, що дозволяє відображати результати обчислень у вигляді 2D/3D графіків, діаграм та функціональних залежностей.
7. Serial – для зв'язку з попередньо запрограмованою платами мікроконтролерів через послідовний порт, що дозволяє виконувати команди без залучення сторонніх програмних платформ;

Детальний текст програмної реалізації алгоритму наведено в додатку В.

Програмна частина платформи складається з двох компонентів: GUI-клієнта (СПМ GUI) та сервера (СПМ API Server).

Архітектура передбачає створення класу $GetAngles$, що інкапсулює основну функціональність.

Основними компонентами класу $GetAngles$ є:

– завантаження та оновлення конфігураційних даних, що включає зберігання координат точок платформи (C, C_2, C_3), середніх ланок (B, B_2, B_3), а також кутів (α, β, γ та a, a_2, a_3). Дані поділені на $zero$ (початкові значення), $last$ (останній стан) та all (історія змін);

– обчислення нових координат платформи за допомогою матриць повороту навколо осей X, Y, Z;

– генерація та відбір точок B, B_2, B_3 з використанням функції get_m_cords, що реалізує розв'язання системи нелінійних рівнянь, які описують геометричні та кінематичні обмеження механізму.

Для фільтрації отриманих кандидатів використовується функція $filter\_points\_on\_circle$, яка перевіряє належність точок колу з радіусом R_1 з урахуванням допустимої похибки ε. Кожна точка, для якої модуль різниці між евклідовою нормою вектора та радіусом R_1 не перевищує ε, вважається такою, що задовольняє умові належності.

Вибір оптимальної комбінації виконується за допомогою функції $choose\_closest\_combo$, яка здійснює пошук комбінації, що мінімізує сумарну евклідову відстань до попередніх координат $last$ та може бути описана за допомогою формули:

$$combo^* = \arg^{min}_{combo} \Sigma_{i=1}^{3} \left\| B_i - B_i^{last} \right\|. \quad (3.7)$$

У вигляді програмного коду реалізація цієї функції наведена в додатку В.

Після визначення координат точок B, B_2, B_3 для кожної пари (B^{last}, B^{new}) формується вектор зміщення та обчислюється кут повороту серводвигуна за формулою (2.20). В результаті отримуються кути a, a_2, a_3, які є безпосередніми керуючими сигналами для серводвигунів.

Візуалізація результатів для 2D та 3D варіантів відбувається на допомогою таких функцій:

– $plot\_combinations$ – відображення можливих варіантів точок у двовимірному просторі. Результати візуалізації наведені на рис. 3.4, де схематично зображено точки C та варіанти точок B на колі.

– $plot\_mechanism\_3d$ – побудова повної тривимірної моделі з урахуванням платформи, точок B, B_2, B_3, центру O та кола платформи з центром у точці P наведена на рис. 3.5, де платформа зображена у вигляді трикутника C, C_2, C_3, точки B, B_2, B_3 у площині XY.

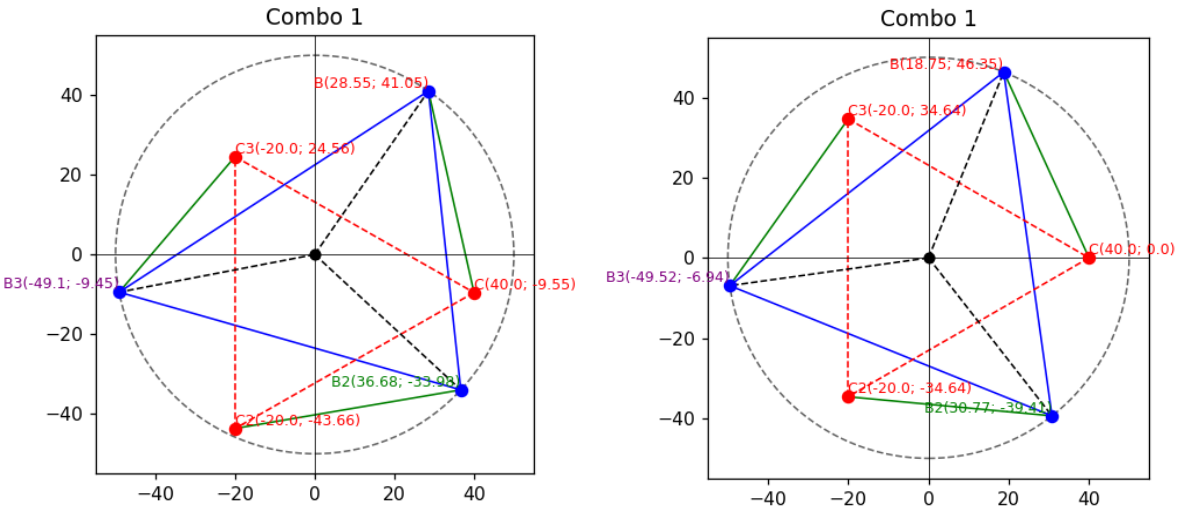


Рис. 3.4 – Приклад 2D візуалізації можливих точок B, B_2, B_3

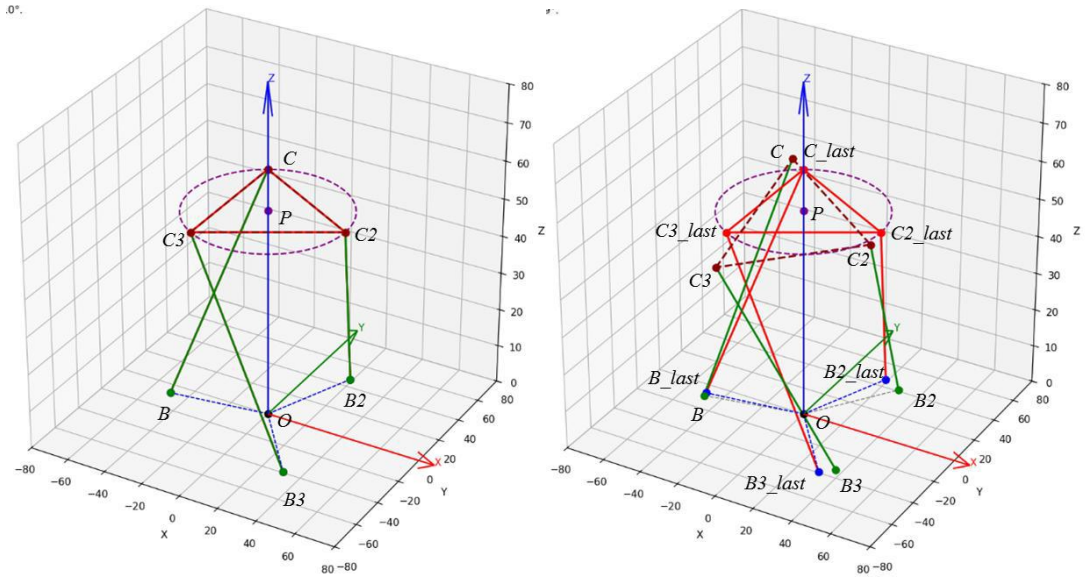


Рис. 3.5 – Приклад 3D візуалізації механізму

Візуалізація у дво- та тривимірному просторі є важливим етапом аналізу та перевірки результатів математичного моделювання сферичних паралельних механізмів. Її використання дозволяє не лише спростити процес налагодження алгоритмів керування та перевірки коректності обчислювальних процедур, але й забезпечує наочне представлення просторової поведінки механізму в різних режимах роботи.

Таким чином, використання засобів візуалізації суттєво підвищує інформативність результатів моделювання, сприяє інтуїтивному розумінню принципів роботи механізму та дозволяє оперативно оцінювати коректність отриманих обчислень і адекватність побудованих алгоритмів керування.

В задачах дистанційного позиціонування камер, виконання операцій у потенційно небезпечних або важкодоступних середовищах (наприклад, під час розмінування, роботи з токсичними чи вибухонебезпечними речовинами, а також при проведенні експериментальних досліджень), а також у системах колаборативної взаємодії людини з технічними засобами [20, 94] доцільним є застосування підходів, що базуються на відстеженні рухів користувача або цільових об’єктів.

У таких умовах особливої актуальності набувають методи комп’ютерного зору, які забезпечують автоматичне виділення та аналіз просторових характеристик об’єктів у полі зору камери. Перспективним напрямком розвитку подібних систем є використання алгоритмів штучного інтелекту [64, 99] для обробки відеопотоку в режимі реального часу, що дозволяє виконувати оцінювання положення та орієнтації об’єктів без необхідності застосування контактних датчиків і додаткових механічних засобів.

Для керування СПМ в режимі реального часу розроблено програмно-апаратну платформу, що використовує інтелектуальну обробку відеоданих для відстеження зміни положення об'єкта керування (голови людини) за допомогою камери та комп'ютера [72, 73]. Такий підхід дозволяє забезпечити інтуїтивно зрозуміле дистанційне керування механізмом без необхідності використання спеціального обладнання.

GUI-клієнт реалізує графічний інтерфейс для керування процесом трекінгу та відображає відеопотік з камери в реальному часі, а сервер виконує обчислення оберненої кінематики на основі отриманих кутів орієнтації.

Взаємодія між компонентами здійснюється через REST API: створення сесії розрахунку з параметрами модуля ai_rtt; відправка кутів з тілом запиту $\{"alpha": \alpha, "beta": \beta, "gamma": \gamma\}$.

Обробка відеопотоку виконується на стороні клієнта з використанням бібліотеки $MediaPipe$. Залежно від вибраного джерела керування використовується один з двох контролерів $HandToPlatformController$ або $FaceToPlatformController$. Контролери реалізують однаковий інтерфейс $process\_frame$, який повертає кадр з намальованими ключовими точками та обчислені кути орієнтації (α, β, γ).

$HandToPlatformController$ Для відстеження руки використовуються три контрольні точки $MediaPipe\ Hands$: точка 0 (зап'ястя), точка 9 (середня фаланга середнього пальця), точка 8 (кінчик вказівного пальця).

$FaceToPlatformController$. Для відстеження обличчя використовуються ключові точки $MediaPipe\ Face\ Mesh$: центр очей (точки 33 та 263), кінчик носа (точка 1), підборіддя (точка 152).

Виконання програми починається з ініціалізації камери та відображення відеопотоку на екрані. При потраплянні руки в кадр бібліотека OpenCV фіксує об'єкт, а бібліотека $MediaPipe$ позначає контрольні точки на знайденому об’єкті. Для визначення просторової орієнтації об’єкта в режимі реального часу використовується підхід на основі бібліотеки MediaPipe, яка забезпечує виділення ключових анатомічних точок руки або обличчя та їх подальшу трекінг-стабілізацію. На основі просторових координат обраних опорних точок формується система векторів, що описують орієнтацію об’єкта у тривимірному просторі. Далі обчислюються кути орієнтації (α, β, γ) як функції відносних положень цих векторів. Отримані кути проходять послідовну обробку, яка включає експоненційне згладжування для зменшення впливу високочастотних тремтінь, калібрування нульової позиції, під час якого положення об'єкта при першому виявленні приймається за початкове, а всі подальші кути визначаються

відносно нього, а також масштабування та обмеження відповідно до заданих масштабних коефіцієнтів і граничних значень. Крім того, для забезпечення дзеркально симетричного керування у випадку використання лівої руки виконується інверсія кутів α та γ.

Відфільтровані кути (α, β, γ) передаються на сервер через REST API із заданим інтервалом (за замовчуванням 2.5 с). На сервері виконується обчислення оберненої кінематики через сервіс $calc\_rtt$, який спочатку відновлює стан сесії включаючи попередні координати точок платформи (C, C_2, C_3), попередні координати проміжних шарнірів (B, B_2, B_3).), попередні кути сервоприводів (a, a_2, a_3). Після цього виконується виклик методу $GetAngles.calculation(\alpha, \beta, \gamma)$, який реалізує повний цикл оберненої задачі задачі кінематики — поворот точок платформи за допомогою матриці $R(\alpha, \beta, \gamma)$, визначення положення проміжних шарнірів через перетин кіл, фільтрацію допустимих конфігурацій, обчислення кутів сервоприводів. Отримані результати разом з оновленим станом сесії зберігаються в базі даних, після чого клієнту повертаються розраховані значення кутів сервоприводів (a, a_2, a_3). Такий підхід забезпечує безконтактне, інтуїтивне та стійке до шумів керування СПМ у реальному часі. Програмна реалізація серверної обробки $calculate$ – наведено в додатку В.

Запропонована архітектура системи забезпечує модульність та масштабованість програмної реалізації, що дозволяє легко інтегрувати нові методи відстеження об’єктів або альтернативні моделі обчислення орієнтації. Завдяки розподіленню обчислювального навантаження між клієнтською та серверною частинами досягається підвищення продуктивності системи та зменшення затримок при обробці даних у реальному часі. Використання уніфікованого API інтерфейсу спрощує взаємодію між модулями та забезпечує незалежність компонентів системи. Реалізація алгоритмів комп’ютерного зору на стороні клієнта мінімізує навантаження на сервер та дозволяє ефективно використовувати ресурси локального пристрою. Запропонований підхід забезпечує гнучкість конфігурації системи під різні сценарії керування та

підвищує надійність і практичну придатність системи для застосування в реальних умовах експлуатації.

Для наочності інтерпретації отриманих результатів на рис. 3.6 наведено відповідні положення руки (обличчя), для яких були отримані окремі значення параметрів, представлені в табл. 3.1.

Результати роботи програми наведено в табл. 3.1.

Табл. 3.1 – Вибірка результатів модуля інтелектуальної обробки даних

Номер вимірювання	Значення повороту платформи	Значення повороту серводвигунів
1	$\alpha = 46°$, $\beta = 43.7°$, $\gamma = 126.9°$	$a = 175.6°$, $a_2 = 116.2°$, $a_3 = 143.8°$
2	$\alpha = 46°$, $\beta = 46°$, $\gamma = 139°$	$a = 175.4°$, $a_2 = 126.6°$, $a_3 = 156.8°$
3	$\alpha = 46°$ $\beta = 46°$, $\gamma = 132.7°$	$a = 178.3°$, $a_2 = 120.3°$, $a_3 = 150.5°$
4	$\alpha = 46°$ $\beta = -3.7°$, $\gamma = 140.9°$	$a = 38°$, $a_2 = 140°$, $a_3 = 118.8°$
5	$\alpha = -1.4°$ $\beta = -3°$, $\gamma = 0.5°$	$a = 49.2°$, $a_2 = 130.5°$, $a_3 = 141.7°$
6	$\alpha = -11°$ $\beta = 12.6°$, $\gamma = 13.7°$	$a = 30.5°$, $a_2 = 88.4°$, $a_3 = 156.1°$

а)

б)

в)

г)

г)

д)

Рис. 3.6 – Наочна інтерпретація вибірки результатів модуля інтелектуальної обробки даних

3.4. Апаратна частина програмно-апаратної платформи

Апаратна частина платформи реалізована на основі мікроконтролера Raspberry Pi Pico RP2040 [63], який виконує функції керування серводвигунами, зчитування даних з інерціального сенсора, взаємодії з користувачем через дисплей та клавіатуру, а також зберігання конфігураційних даних на microSD-карті. Структурну схему апаратної частини наведено на рис. 3.7.

Мікроконтролер Raspberry Pi Pico [61] побудовано на базі двобічного мікропроцесора RP2040 з тактовою частотою 133 МГц, 264 КБ статичної пам'яті та 2 МБ флеш-пам'яті. Він підтримує необхідний набір периферійних інтерфейсів: I2C, SPI, UART, GPIO та ШІМ [68, 69].

Для вимірювання орієнтації платформи використовується мікросхема MPU6050 [57, 58, 93] , яка об'єднує триосьовий акселерометр, триосьовий гіроскоп. Акселерометр має вибірні діапазони ±2, ±4, ±8, ±16 g, гіроскоп — ±250, ±500, ±1000, ±2000 °/с. З'єднання з мікроконтролером виконано через інтерфейс I2C1 на частоті 400 кГц. Дані MPU6050 використовуються в комплементарному фільтрі для отримання стабільної оцінки кутів крену та тангажу.

Безпосереднє приведення у рух платформи сферичного паралельного механізму забезпечують три серводвигуни [65], які керуються через ШІМ-сигнали з частотою 50 Гц та тривалістю імпульсу від 500 до 2500 мкс. Для кожного серводвигуна виділено окремий вивід ШІМ. Діапазон кутів повороту серводвигунів становить 0–180°. Для живлення серводвигунів використовується окреме джерело живлення 5 В з максимальним струмом до 2 А на кожен двигун.

Для взаємодії з користувачем у автономному режимі (без підключення до комп'ютера) використовуються OLED-дисплей SSD1306 з роздільною здатністю 128×64 пікселів та матрична клавіатура 4×4. Дисплей підключено через інтерфейс I2C0 на частоті 400 кГц, для відображення поточних значень кутів платформи, кутів сервоприводів, даних з MPU6050 та меню налаштувань. Клавіатуру підключено за схемою матричного сканування. Клавіатура містить 16 клавіш: цифри 0–9, символи A–D, *, #.

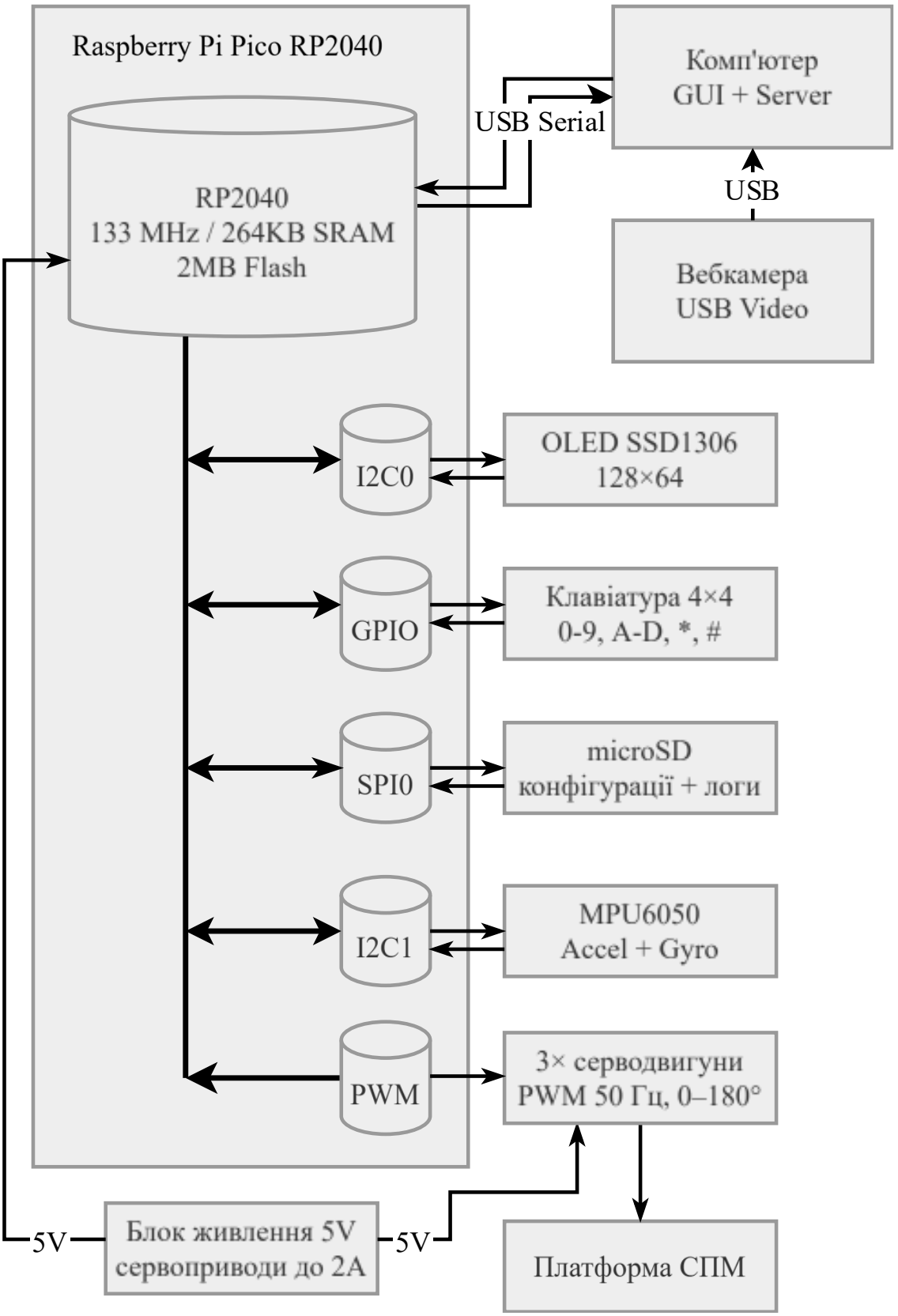


Рис. 3.7 – Структурна схема апаратної частини

Для зберігання конфігураційних файлів (системної конфігурації СПМ, кутів та координат) використовується microSD-карта, підключена через інтерфейс SPI0. Частота SPI становить 400 кГц під час ініціалізації та 4 МГц у робочому режимі. Конфігураційні файли зберігаються у форматі JSON.

Для реалізації режиму керування через відстеження руки або обличчя використовується вебкамера, підключена до комп'ютера через USB Video. Захоплення відео виконується за допомогою OpenCV на стороні GUI-клієнта, після чого кадри передаються на обробку контролерам MediaPipe.

Механічна частина являє собою сферичний паралельний механізм [26] із трьома кінематичними ланцюгами. Кожен ланцюг складається із серводвигуна, проміжного шарніра та точки кріплення на платформі. Механізм може працювати у двох конфігураціях: правій (CW) та лівій (CCW) залежно від напрямку згину.

Платформа живиться від зовнішнього блока живлення 5 В. Raspberry Pi Pico може отримувати живлення як через USB-з'єднання з комп'ютером, так і від зовнішнього джерела, що дозволяє працювати як у складі програмно-апаратного комплексу, так і автономно.

3.5. Висновки до розділу 3

Запропоновано метод ідентифікації параметрів руху елементів сферичного паралельного механізму на основі використання алгоритму комплементарного фільтра, адаптованого для систем керування СПМ. Комплементарний фільтр потребує мінімальних обчислювальних затрат та є простим у налаштуванні, що важливо для систем керування в реальному часі з обмеженими ресурсами. Розроблений алгоритм поєднує дані з гіроскопа та акселерометра для оцінки кута нахилу рухомої платформи СПМ, зменшуючи вплив дрейфу гіроскопа та шумів акселерометра. Експериментальна перевірка на дослідному зразку СПМ підтвердила працездатність та ефективність запропонованого підходу для ідентифікації кутового положення рухомої платформи. Представлені результати експериментальних досліджень на реальному прототипі СПМ демонструють практичну ефективність запропонованого методу фільтрації для ідентифікації параметрів руху в умовах реального часу.

Для керування СПМ у режимі реального часу розроблено програмно-апаратну платформу, що забезпечує інтелектуальну обробку відеоданих та відстеження положення об’єкта керування за допомогою камери. Система

реалізована у вигляді дворівневої архітектури, де клієнтський графічний інтерфейс виконує захоплення та первинну обробку відеопотоку, а серверна частина відповідає за обчислення оберненої кінематики та формування керуючих сигналів для серводвигунів.

Розроблено програмний модуль визначення положення об’єкта керування на основі класу $GetAngles$, який забезпечує обробку вхідних кутових даних та їх перетворення у керуючі впливи для виконавчих механізмів. Додатково реалізовано засоби візуалізації роботи системи у дво- та тривимірному просторі, що дозволяє контролювати коректність функціонування алгоритмів у режимі реального часу. Запропонований підхід забезпечує узгоджену роботу клієнтської та серверної частин системи, що дозволяє реалізувати дистанційне керування СПМ на основі відеоданих без використання контактних датчиків. Отримані результати підтверджують працездатність програмної реалізації та її придатність для задач керування в реальному часі. Запропонована система забезпечує візуалізацію руху у дво- та тривимірному просторі, що підвищує наочність процесу функціонування та спрощує перевірку коректності роботи алгоритмів. Це дозволяє здійснювати оперативну інтерпретацію результатів обчислень і оцінювати поведінку механізму в режимі реального часу. Двовимірна візуалізація дає можливість аналізувати проекції траєкторій руху, характер зміни координат та кутових параметрів, а також виявляти потенційні ділянки з різкими переходами або нелінійною динамікою. Тривимірне представлення забезпечує повноцінне відображення просторової конфігурації механізму, дозволяючи оцінювати взаємне розташування ланок, характер орієнтації рухомої платформи та дотримання кінематичних обмежень. Отримані результати підтверджують ефективність запропонованих програмних рішень та їх придатність для інтеграції в автоматизовані системи керування СПМ.

РОЗДІЛ 4 ВИКОРИСТАННЯ СФЕРИЧНОГО ПАРАЛЕЛЬНОГО МЕХАНІЗМУ В СИСТЕМАХ АВТОМАТИЧГОЇ ДІАГНОСТИКИ ПРОМИСЛОВИХ ПОВЕРХОНЬ

Практичне впровадження сферичних паралельних механізмів потребує вирішення низки прикладних задач, серед яких ключовими є вибір відповідних вимірювальних засобів, розробка ефективних алгоритмів планування руху та побудова архітектури автоматизованої системи керування. В даному розділі розглянуто практичні аспекти застосування СПМ, наприкладі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів, що є перспективним напрямком використання цих механізмів [23, 24].

Промислові автоклави є критично важливим обладнанням у багатьох галузях, включаючи харчову та хімічну промисловість. Забезпечення надійної та безпечної експлуатації автоклавів є важливим завданням, оскільки не виявлені вчасно пошкодження внутрішніх поверхонь автоклавів можуть призвести до суттєвих економічних втрат, зупинки виробництва та становити загрозу для життя персоналу [100]. Традиційні методи контролю стану внутрішніх поверхонь автоклавів такі як візуальний, ультразвуковий, радіографічний, магнітопорошковий, капілярний контроль та використання методу акустичної емісії [101] часто є трудомісткими, вимагають значного часу на підготовку та проведення діагностики, а також можуть бути суб’єктивними та не завжди забезпечувати повне охоплення всієї поверхні, що призводить до пропуску важливих дефектів, особливо у важкодоступних місцях. Тому розробка та впровадження автоматизованих систем діагностики внутрішніх поверхонь промислових автоклавів є особливо актуальною, оскільки це забезпечить значне підвищення ефективності, швидкості, точності та об’єктивності контролю. Перспективним напрямком у створенні таких автоматизованих систем є використання сферичних паралельних механізмів. Завдяки кінематичній структурі, СПМ забезпечують високу маневреність та можливість орієнтації пристрою або сенсору, який використовується для збору інформації про стан

внутрішньої поверхні, в широкому діапазоні кутів, що є особливо важливим для дослідження поверхонь автоклавів зі складними геометричними формами.

4.1. Методи сканування поверхонь автоклавів

Різноманітність сучасних методів сканування надає технічну можливість отримати детальні дані про стан внутрішньої поверхні автоклаву. Щоб визначити найбільш ефективні підходи для виявлення дефектів проаналізуємо найбільш поширені методи сканування, кожен з яких має свої переваги та недоліки [24].

Лазерне сканування [102] використовує лазерний промінь для вимірювання відстані до поверхні об’єкта. Сканер випромінює лазерний промінь, який відбивається від поверхні та повертається до сканера. Час, який потрібен лазерному променю для переміщення до поверхні та у зворотному напрямку, використовується для обчислення відстані. Такий метод сканування має високу точність, швидкість сканування, роздільну здатність та придатний для сканування складних поверхонь. Проте має обмеження у скануванні дзеркальних та прозорих поверхонь та чутливий до зовнішніх умов (освітлення, пил).

Фотограмметрія [103] аналізує серію фотографій об’єкта для створення тривимірної моделі. Фотографії робляться з різних кутів, а потім обробляються за допомогою спеціального програмного забезпечення. Програмне забезпечення використовує перекриття між фотографіями для обчислення тривимірних координат точок на поверхні об’єкта. Використання цього методу надає можливість сканування великих об’єктів з високим ступенем деталізації при відносно низькій вартості обладнання. До недоліків методу можна віднести потребу у якісному освітленні та тривалий процес обробки даних. Використання сферичних паралельних механізмів [34] може значно полегшити та покращити процес фотограмметричного сканування внутрішньої поверхні автоклаву, особливо якщо доступ до внутрішньої частини обмежений або потрібна точна орієнтація камери.

В промисловості використовуються також ультразвуковий контроль, рентгенівське сканування, вихрострумовий контроль, магнітопорошковий контроль, капілярний контроль та інші методи [104].

Вибір оптимального методу сканування залежить від конкретних вимог до точності, швидкості та вартості. Однак, ефективність виявлення дефектів залежить не лише від методу сканування, але й від методів обробки отриманих даних. Після сканування дані обробляються за допомогою спеціалізованого програмного забезпечення. Процес обробки даних може включати такі етапи як фільтрація шуму, вирівнювання та згладжування поверхні, сегментація даних, аналіз отриманих зображень, виявлення дефектів.

4.2. Методи планування траєкторій сканування на основі геометричних моделей

Розробка ефективних методів автоматизованого сканування внутрішньої поверхні автоклава з урахуванням складності форми є важливим завданням при реалізація систем реального часу для керування рухом приводів СПМ з метою точного позиціонування та орієнтації діагностичного інструменту у визначених точках траєкторії. Розробка алгоритмів планування траєкторії для автоматичного визначення оптимального шляху руху діагностичного інструменту всередині автоклава включає врахування складної геометрії внутрішньої поверхні, наявності перешкод, внутрішніх елементів конструкції, необхідності повного покриття досліджуваної області та мінімізації часу діагностики. Для планування траєкторії руху діагностичного інструменту, встановленого на сферичному паралельному механізмі, можуть застосовуватися методи на основі геометричних моделей та алгоритми пошуку шляху [105].

Методи на основі геометричних моделей [103] передбачають створення точної цифрової моделі внутрішнього простору автоклава, включаючи його геометричні характеристики та розташування можливих перешкод. На основі таких моделей використовуються різні геометричні алгоритми для визначення шляху, вільного від перешкод. Наприклад, робочий простір розбивається на

тривимірні кубічні елементи – вокселі. Вокселі позначаються як вільні або зайняті перешкодами. Планування шляху полягає у знаходженні послідовності вільних вокселів між початковою та кінцевою точками. Ще один підхід полягає в моделюванні не фізичного простору, а простору можливих конфігурацій маніпулятора, наприклад, кутів поворотних ланок. Перешкоди у фізичному просторі відображаються як заборонені області в конфігураційному просторі, в якому відбувається планування шляху. Для СПМ конфігураційний простір може бути складним через його кінематику. Методи на основі геометричних моделей дозволяють точно враховувати геометрію робочого простору і перешкод, а також можуть забезпечити оптимальні або близькі до оптимальних шляхи, залежно від використовуваного алгоритму. Проте, побудова точної геометричної моделі може бути складною, особливо для автоклавів складної форми або з невідомими внутрішніми елементами, а обчислювальна складність алгоритмів може зростати експоненційно зі збільшенням розмірності робочого простору.

Алгоритми пошуку шляху використовуються для знаходження оптимального або допустимого шляху між початковою та кінцевою точками в графі, де вузлами є можливі стани, наприклад, положення та орієнтація інструменту, а ребрами – можливі переміщення між цими станами. Прикладом таких алгоритмів є A^* – евристичний алгоритм пошуку найкоротшого шляху [99], який використовує функцію оцінки

$$f(n) = g(n) + h(n), \tag{4.1}$$

де $g(n)$ – вартість шляху від початкової точки до поточного вузла n, $h(n)$ – евристична оцінка вартості шляху від вузла n до цільової точки. Вузлами графа можуть бути дискретизовані положення та орієнтації інструменту, а вартість ребер – відстань або час переміщення. RRT – ймовірнісний алгоритм [21, 62, 106], який досліджує простір конфігурацій, будуючи дерево випадкових станів, починаючи з початкової конфігурації. Нові стани генеруються випадково, після чого до дерева додається найближчий існуючий стан, якщо з’єднання з новим станом не призводить до зіткнення. Пошук шляху завершується, коли дерево

досягає цільової конфігурації. Алгоритм RRT та його модифікації підходить для задач з високою розмірністю конфігураційного простору та складними перешкодами. Наведені алгоритми пошуку шляху добре опрацьовані та мають теоретичне обґрунтування, проте дискретизація робочого або конфігураційного простору може призвести до втрати точності або збільшення обчислювальної складності, ефективність алгоритму A* істотно залежить від якості евристичної функції, а RRT є ймовірнісним алгоритмом і не завжди гарантує знаходження оптимального шляху.

Вибір конкретних методів або їх комбінацій залежить від складності геометрії автоклава, наявності перешкод, вимог до повноти покриття, точності позиціонування та часу діагностики.

4.3. Вибір стратегії сканування внутрішній поверхні автоклава

Стратегія сканування визначає спосіб переміщення діагностичного інструменту по внутрішній поверхні автоклава для її повного дослідження. Вибір стратегії сканування починається з розділення внутрішньої поверхні автоклава на скінченну кількість точок або областей для проведення вимірювань або збору даних датчиком. Кількість та розташування цих точок визначають щільність сітки сканування, де щільніша сітка забезпечує детальніше покриття та вищу ймовірність виявлення дрібних дефектів, проте збільшує час сканування та обсяг даних для подальшої обробки. У деяких випадках доцільно використовувати адаптивну сітку, щільність якої змінюється залежно від попередніх даних або характеристик поверхні, збільшуючись в областях з підозрілими ознаками. Способи формування сітки залежать від форми автоклава: лінійне сканування передбачає рух інструменту паралельними лініями, що забезпечує покриття всієї поверхні, однак може бути неефективним для складних геометричних форм. Спіральне сканування полягає в русі інструменту по спіралі з наближенням до центру або віддаленням від нього, що може бути ефективним для циліндричних або сферичних поверхонь. Сканування за контуром передбачає рух інструменту вздовж заданих контурів або перерізів поверхні. Для автоклавів складної форми

може знадобитися складніша параметризація поверхні та проєктування для неї сітки.

Форма автоклава значною мірою визначає оптимальну траєкторію руху діагностичного інструменту. При автоматизованій діагностиці внутрішніх поверхонь найчастіше застосовується рух діагностичного інструменту по траєкторіях: спіраль, коло, зигзаг, квадрат [24, 107], що обумовлюється їх оптимальними характеристиками для заданих форм автоклавів, типів датчиків та завдань діагностики. Приклади формування таких траєкторій для різних форм автоклавів та їх елементів наведено на рис. 4.1.

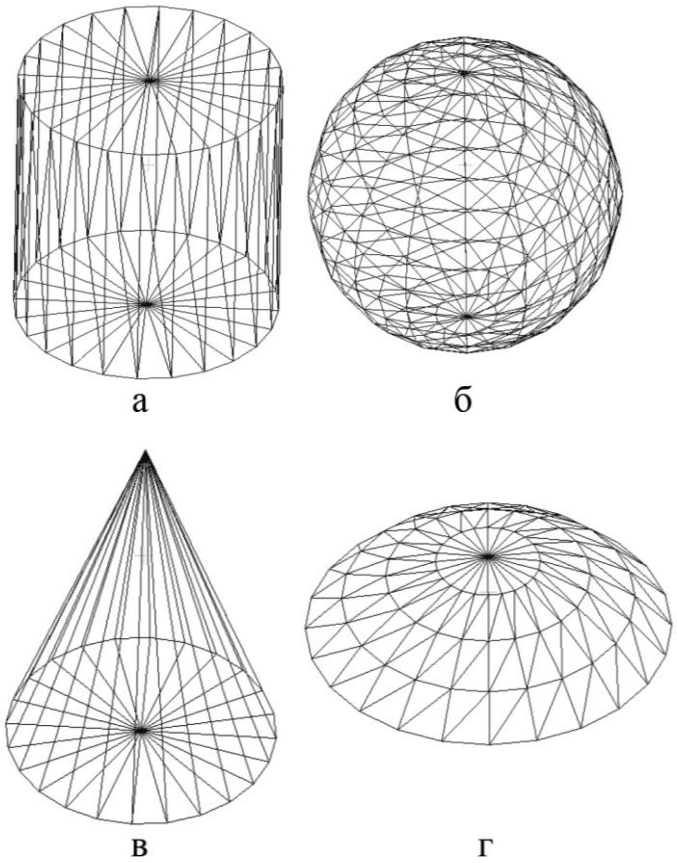


Рис. 4.1 – Приклади формування траєкторій сканування для різних форм автоклавів та їх елементів

Спіральна траєкторія ефективна для обстеження поверхонь з осьовою симетрією, таких як циліндричні або конічні частини автоклавів, а також сферичні днища чи кришки, оскільки забезпечує поступове та рівномірне покриття поверхні, починаючи з центральної точки та розширюючись до краю

елементу, що сканується. Такий підхід забезпечує рівномірне покриття та відносно однорідну щільність сканування по всій площі, що важливо для виявлення дефектів будь-якого розміру та розташування, а також ефективність для круглих або осьово-симетричних форм, що мінімізує холості переміщення порівняно з лінійним скануванням. Також спіральна траєкторія часто може бути реалізована як безперервний рух, що зменшує навантаження на механізм та скорочує час сканування. Завдяки цим властивостям, спіральне сканування ідеально підходить для обстеження внутрішніх стінок циліндричних автоклавів (при комбінації руху вздовж осі з обертанням) та для сканування сферичних або злегка вигнутих поверхонь.

Кругові траєкторії застосовуються для детального обстеження локалізованих областей або для сканування поверхонь, що мають кругову симетрію в певному перерізі, таких як фланцеві з’єднання, отвори або кільцеві зварні шви. Рух по кругу може використовуватися для підтримки постійної відстані або кута між датчиком та поверхнею при обстеженні циліндричних або конічних об’єктів шляхом руху навколо їхньої осі на фіксованій висоті. Кругові траєкторії надають можливість детального обстеження локальних зон завдяки багаторазовому проходженню однієї й тієї ж області для глибшого аналізу, забезпечують спрощення підтримки постійних параметрів сканування, таких як фіксована відстань або кут між датчиком і поверхнею, що є важливим для деяких типів датчиків. Ще однією перевагою є простота програмування руху навколо циліндричних або круглих елементів конструкції.

Зигзагоподібна траєкторія – це варіант лінійного сканування, що застосовується для покриття плоских або близьких до плоских поверхонь, а також для сканування розгорток циліндричних поверхонь. Зміна напрямку руху на кінці кожної лінії забезпечує ефективне покриття всієї області без необхідності повернення до початкової точки кожної нової лінії. Сканування по зигзагоподібній траєкторії забезпечує повне покриття плоских областей при відповідній програмній реалізації послідовності лінійних рухів зі зміною напрямку. Застосовується зигзагоподібне сканування для обстеження плоских

днищ або кришок автоклавів, а також циліндричних стінок, за умови можливості лінійного переміщення датчика вздовж та навколо осі циліндра.

Квадратна або прямокутна траєкторія використовується для покриття плоских або локалізованих областей при детальному скануванні невеликих ділянок з метою виявлення локальних дефектів або для калібрування датчика на певній ділянці поверхні, а також як базова комірка для складніших стратегій сканування великих площ.

Перевагами такого підходу є детальне обстеження невеликих зон, систематичне покриття локальних прямокутних областей та простота програмування послідовності рухів по периметру квадрата або прямокутника із заданим кроком для локальних завдань. Застосовується для детального сканування областей навколо виявлених дефектів, зон підвищеної корозійної активності, а також для калібрувальних процедур на тестових ділянках.

Програмна реалізація алгоритму визначення положення платформи, описана в підрозділі 3.3, може бути використана для керування сферичним паралельним механізмом в складі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів у двох ключових сценаріях. По-перше, запропонована архітектура на основі інтелектуальної обробки відеоданих забезпечує інтуїтивне безконтактне керування орієнтацією діагностичного інструменту у випадках, коли параметри автоклаву відхиляються від номінальних через деформації, зношування чи конструктивні невідповідності — оператор може безпосередньо задавати положення платформи через відстеження руки або обличчя, що дозволяє адаптувати траєкторію сканування до реальних умов без необхідності побудови попередньої цифрової моделі. По-друге, при виявленні попередніми методами аналізу хмар точок потенційних дефектів описаний підхід дозволяє оперативно переміщувати діагностичний інструмент до підозрілих ділянок для детального підтверджувального сканування, забезпечуючи локальне обстеження з високою точністю позиціонування та зміною кута огляду в реальному часі. Таким чином, програмна реалізація алгоритму визначення положення платформи на основі відеотрекінгу розширює функціональні можливості автоматизованої системи діагностики, надаючи як

резервний режим керування при невідомих або змінених параметрах об'єкта, так і засіб цільової верифікації виявлених дефектів.

Використання сферичного паралельного механізму в структурі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів є перспективним, що обумовлюється його високою маневреністю, здатністю орієнтувати діагностичний інструмент у широкому діапазоні кутів [24, 26].

Структура СПМ з кількома ступенями вільності обертального руху дозволяє ефективно досягати та сканувати важкодоступні ділянки, забезпечуючи повніше покриття поверхні порівняно з традиційними послідовними маніпуляторами, які можуть мати обмеження в орієнтації скануючого інструменту.

Застосування СПМ дозволяє реалізувати різні стратегії сканування, включаючи спіральне, кругове, лінійне або комбіновані траєкторії, адаптовані до конкретної форми автоклава та вимог до обстеження. Можливість точного програмування рухів СПМ сприяє автоматизації процесу сканування, зменшуючи час діагностики та вплив людського фактора, підвищуючи тим самим ефективність та об’єктивність контролю стану внутрішньої поверхні автоклавів.

Оскільки ефективність запропонованої системи безпосередньо залежить від кінематичних можливостей сферичного паралельного механізму, у подальшому аналізі використовується його математична модель, опис якої наведено в розділі 2.

4.4. Отримання та побудова хмари точок за даними фотограмметричного сканування

Використання хмар точок в задачі автоматизованого сканування поверхні автоклаву є перспективним сучасним підходом, що надає низку вагомих переваг, які роблять цей метод особливо ефективним для виявлення дефектів [31]. Хмари точок – це набори точок у тривимірному просторі, які представляють собою геометричну поверхню об’єкта або простору. Хмари точок забезпечують

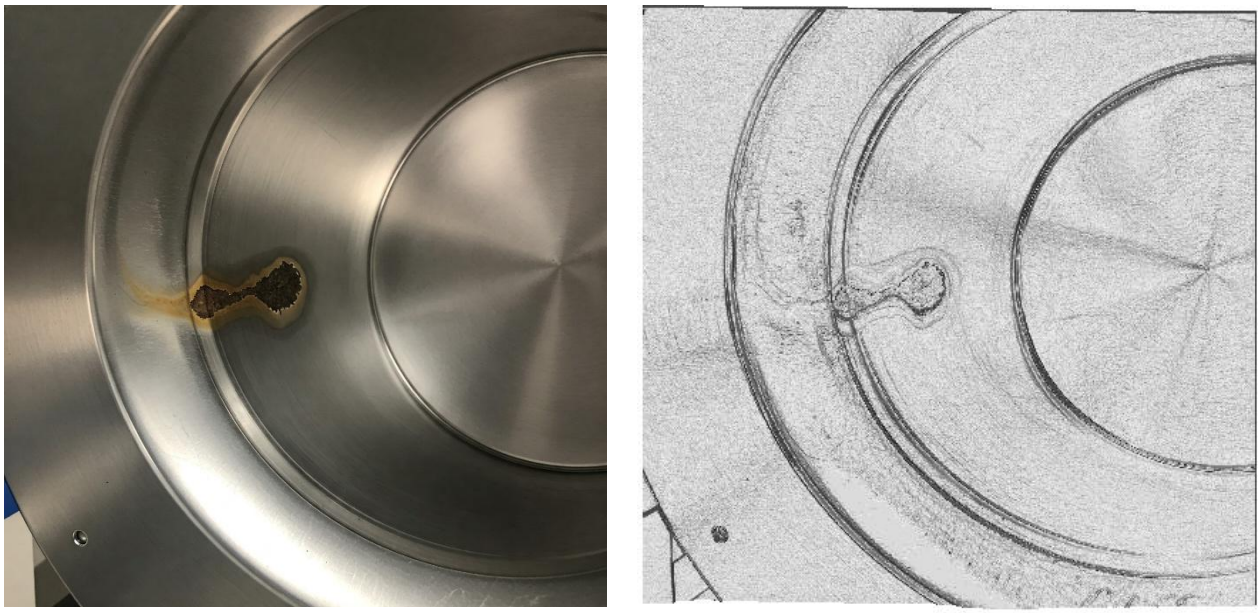
надзвичайно високу точність вимірювань, що дозволяє детально відтворити навіть найдрібніші дефекти поверхні автоклаву, що є особливо важливо для виявлення тріщин, подряпин, корозії та інших дрібних дефектів, які можуть бути непомітними при використанні інших методів. При цьому отримується повна тривимірна інформація про поверхню автоклаву, що дозволяє аналізувати не тільки поверхневі дефекти, але й їх глибину та форму. Методи сканування, що використовуються для створення хмар точок, зазвичай є неруйнівними і не пошкоджують поверхню автоклаву під час сканування. Обробка хмар точок може бути автоматизована за допомогою спеціалізованого програмного забезпечення, що дозволяє швидко та ефективно аналізувати великі обсяги даних та виявляти дефекти. Хмари точок дозволяють візуалізувати результати сканування у тривимірному просторі, що полегшує аналіз дефектів та їх локалізацію та сприяє використанню алгоритмів машинного навчання для виявлення дефектів. Алгоритми машинного навчання можуть бути навчені розпізнавати різні типи дефектів на основі характеристик хмари точок, що дозволяє автоматизувати процес класифікації дефектів та підвищити точність їх виявлення.

Для побудови хмар точок по фото об’єкту існує різне спеціалізоване програмне забезпечення, що використовує алгоритми фотограмметрії, наприклад: RealityCapture відоме своєю швидкістю та ефективністю в обробці великих обсягів фотографій; 3DF Zephyr надає широкий спектр інструментів для фотограмметрії, включаючи побудову хмар точок та тривимірних моделей; Meshroom безкоштовне та відкрите програмне забезпечення для фотограмметрії.

Основні алгоритми, які використовуються у фотограмметричному програмному забезпеченні для побудови хмар точок з фотографій це алгоритм структурного відновлення руху (Structure from Motion, SfM) [21], який аналізує серію перекриваючихся фотографій для визначення положення камери та орієнтації фотографій та виявляє спільні точки на фотографіях та використовує їх для розрахунку тривимірних координат точок. Ще один алгоритм – багатовимірне стереозображення (Multi-View Stereo, MVS) [108] – використовує інформацію про положення камери та орієнтацію фотографій, отриману з SfM,

для створення щільної хмари точок та аналізує перекриття між фотографіями для визначення тривимірних координат точок з високою точністю.

Ці програми та алгоритми дозволяють отримувати точні та детальні хмари точок з фотографій, що можна використати для обробки зображень внутрішньої поверхні автоклаву. На рис. 4.2. наведено фото автоклаву зі слідами корозії та відповідна йому згенерована хмара точок.



а) б)

Рис. 4.2 – Автоклав зі слідами корозії:
а) фото фрагменту, б) згенерована хмара точок

4.5. Алгоритм обробки хмари точок за даними фотограмметричного сканування

Для розрахунку нормалей для кожної точки хмари точок використаємо алгоритм головних компонент (PCA) [109, 31]. Для кожної точки хмари визначимо її сусідні точки, використовуючи алгоритм k - найближчих сусідів [110]. Це алгоритм машинного навчання, який базується на припущенні, що схожі об’єкти знаходяться близько один до одного.

Алгоритм k - найближчих сусідів складається з наступних етапів:

1. Визначається кількість найближчих сусідів k, які будуть використовуватися для прийняття рішення. Вибір k істотно впливає на результати алгоритму, адже мале значення робить модель чутливою до шуму та викидів, а велике – згладжує рішення, що може призвести до втрати локальних особливостей та деталей сканованого об'єкта. Дрібні дефекти або нерівності на поверхні можуть бути проігноровані, оскільки вони не будуть достатньо представлені серед великої кількості сусідів, тому важливо вибрати оптимальне значення k, яке забезпечить баланс між згладжуванням рішень і локальною чутливістю.

2. Обчислюється відстань між новою точкою даних x і кожною точкою даних x_i з навчального набору.

$$d(x, x_i) = \sqrt{\sum \left(x_j - x_{ij}\right)^2}, \tag{4.2}$$

де x – нова точка даних, x_i – точка даних з навчального набору, x_j – j-та координата x, x_{ij} – j-та координата x_i.

3. Вибираються k точок даних з навчального набору, які мають найменшу відстань до нової точки даних.

Для кожної точки та її сусідніх точок будуємо коваріаційну матрицю C, яка описує розсіювання точок у просторі. Коваріаційна матриця обчислюється за формулою:

$$\mathrm{C} = \frac{1}{k-1}(A-\mu)^T(A-\mu), \tag{4.3}$$

де k – задана кількість сусідніх точок, A – матриця сусідніх точок, розмірністю $[k \times 3]$, μ – вектор середніх значень координат сусідніх точок $[\mu X_c \quad \mu Y_c \quad \mu Z_c]$, який віднімається від кожного рядка матриці A.

Для коваріаційної матриці розрахуємо власні вектори та власні значення. Аналіз власних векторів дає можливість оцінити напрямки головних компонент, а власні значення – дисперсію точок у цих напрямках.

Власні вектори v та власні значення λ коваріаційної матриці обчислюються з рівняння $\mathrm{C}v = \lambda v$, згідно якого власні (характеристичні) значення розраховуються з умови:

$$|\lambda I - C| = 0, \tag{4.4}$$

де I – одинична матриця тієї ж розмірності, що і матриця C.

Нормаль $N = [N_X, N_Y, N_Z]$ до поверхні в точці дорівнює власному вектору, що відповідає найменшому власному значенню. Цей власний вектор вказує напрямок, у якому точки найменше розсіюються, тобто напрямок, перпендикулярний до поверхні. Також необхідно враховувати, що якість розрахунку нормалей залежить від щільності хмари точок. В табл. 4.1 наведено результати розрахунків вибіркових координат нормалей точок хмари, наведеній на Рис. 4.2 б. В процесі аналізу отриманої хмари точок було проведено фільтрацію у декілька етапів. Оригінальна хмара точок, яка налічувала 100162 точок неведена на рис. 4.3

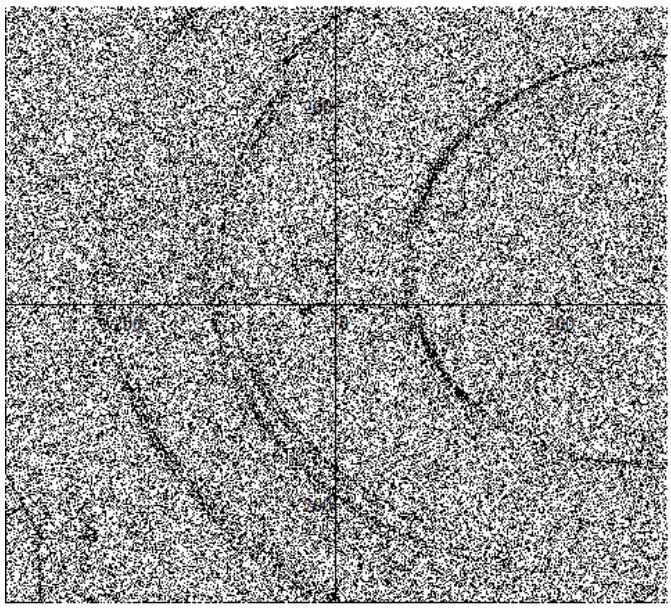


Рис. 4.3 – Початкова хмара точок: *n*=100162

Результати етапів обробки, представлені на площині XOY, та кількість точок n, що залишилася у хмарі точок після кожного етапу, наведені на рисунках з рис. 4.4 по рис. 4.8.

Табл. 4.1 Результати обчислень координат нормалей точок хмари

№ точки у хмарі	Координати точки			Координати нормалі		
	X	Y	Z	N_X	N_Y	N_Z
1	22.518	-289.489	15	$9.78 \cdot 10^{-4}$	$9.78 \cdot 10^{-4}$	1
5000	-162.176	-275.391	-7.083	0.191	$1.145 \cdot 10^{-3}$	0.982
10000	116.444	-217.202	-9.47	$1.145 \cdot 10^{-3}$	0.191	0.982
15000	285.499	-158.799	-2.115	0.225	-0.897	0.38
20000	-59.489	-104.301	9.368	0.881	0.241	0.408
25000	20.756	-50.37	-2.529	$9.78 \cdot 10^{-4}$	$9.78 \cdot 10^{-4}$	1
30000	-30.247	7.148	-11.793	0.269	0.675	0.687
35000	-295.945	64.738	-7.677	0.191	$1.145 \cdot 10^{-3}$	0.982
40000	-255.319	122.229	-8.328	$1.145 \cdot 10^{-3}$	-0.191	0.982
45000	-145.082	180.942	-7.565	-0.191	$1.145 \cdot 10^{-3}$	0.982
50000	-53.512	239.934	-4.546	0.748	-0.635	0.192

На першому етапі з початкової хмари точок були видалені точки в яких всі координати нормалі дорівнюють нулю $[N_X, N_Y, N_Z] = [0,0,0]$, внаслідок чого кількість точок скоротилася до 55850 (рис. 4.4). Такі точки відносяться до ізольованих точок, які не мають достатньої кількості сусідів для визначення локальної поверхні. Якщо алгоритм намагається розрахувати нормаль для ізольованої точки, він може повернути нульовий вектор як ознаку невизначеності. Всі координати нормалі дорівнюють нулю також у випадку виродженої локальної геометрії, якщо сусідні точки співпадають з розглядуваною точкою. У цьому випадку коваріаційна матриця C буде нульовою, і власні вектори v будуть невизначені або нульовими. Ще одним випадком, коли вектор нормалі нульовий – є лінійне розташування точок. Якщо всі сусідні точки

лежать на одній прямій лінії, дотична площина не є однозначно визначеною в тривимірному просторі, і розрахунок нормалі може бути некоректним.

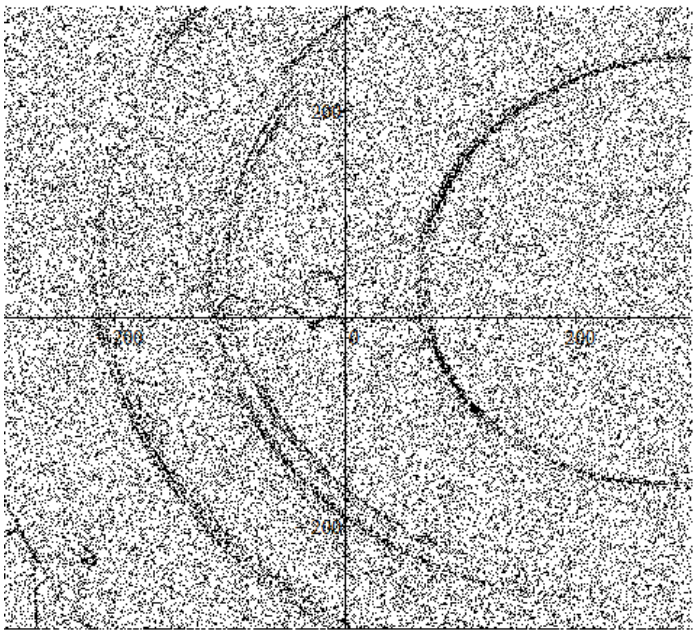


Рис. 4.4 – Результати першого етапу обробки хмари точок: n=55850

На другому етапі вилучені точки, координати нормалі яких мають однакові значення, що вказує на певні характеристики сканованої поверхні, наприклад наявність великих планарних поверхонь (рис. 4.5). Якщо значна частина сканованої поверхні є плоскою, то нормалі до цих точок будуть паралельними або матимуть однакові (або дуже близькі) значення координат. Велика кількість таких точок вказує на те, що значна частина об’єкта може складатися з однієї або кількох великих плоских ділянок, орієнтованих в одному напрямку.

В деяких випадках, особливо при з’єднанні кількох знімків, можуть виникати спотворення, які призводять до того, що велика кількість точок отримує однакові або дуже схожі нормалі. Це може бути пов’язано з помилками реєстрації або калібрування сканера. На цьому етапі вилучаються також точки, що належать симетричним або геометрично простим об’єктам, адже якщо сканований об’єкт має значну симетрію або просту геометричну форму (наприклад, циліндр з плоскою основою), то великі частини поверхні можуть мати однакові нормалі. За результатами цього етапу кількість точок скоротилася до 42900.

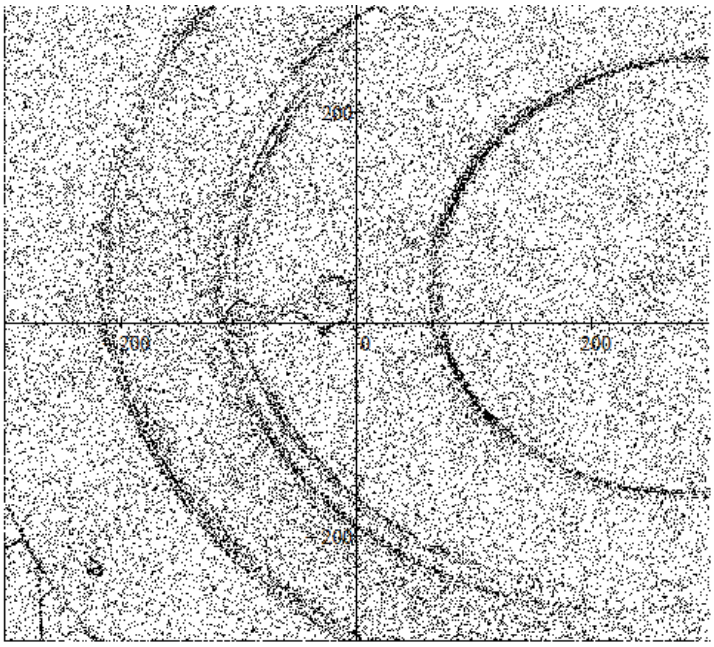


Рис. 4.5 – Результати другого етапу обробки хмари точок: *n*=42900

На третьому етапі видалені точки, координата нормалі яких по осі аплікат Z дорівнюватиме одиниці або приблизно одиниці (рис. 4.6). Якщо N_Z - координата нормалі дорівнює одиниці то локальна поверхня є ідеально плоскою та горизонтальною, паралельною площині XY, і вектор нормалі є одиничним та спрямованим точно вгору.

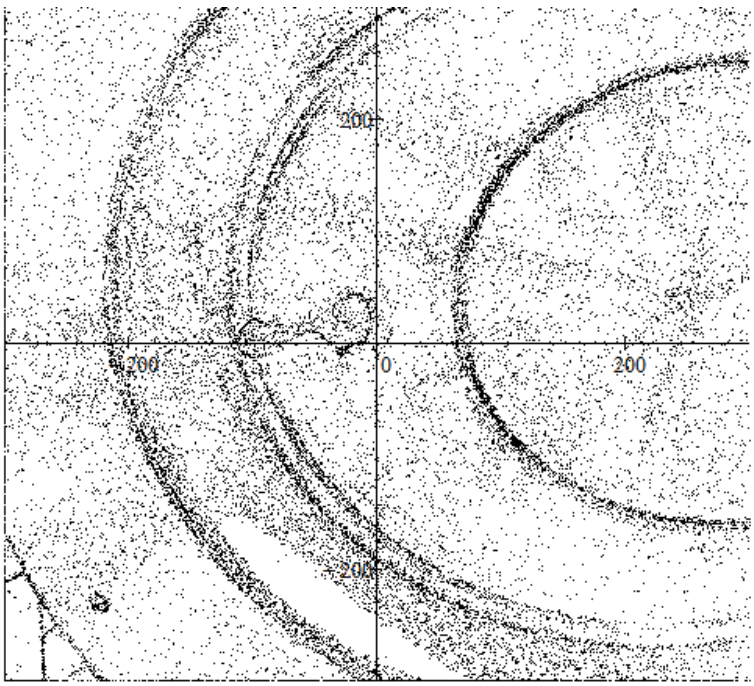


Рис. 4.6 – Результати третього етапу обробки хмари точок: *n*=27690

Більш реалістичний сценарій при скануванні реальних об’єктів спостерігається коли координата нормалі N_Z близька до одиниці. При цьому локальна поверхня може мати невеликі відхилення від ідеальної горизонтальності. Наприклад, трохи нахилена площина або слабо викривлена поверхня, де переважає горизонтальна орієнтація.

На практиці, при локальній апроксимації майже плоских горизонтальних ділянок складної поверхні, якщо локально сусідні точки утворюють майже горизонтальну площину, компонента нормалі N_Z буде близькою до 1.

На четвертому етапі з точок, що залишились у хмарі точок обираємо точки, що мають координату Z менше нуля (рис. 4.7). З точки зору дефектоскопії, якщо координата Z сканованої поверхні менше нуля, це може вказувати на кілька важливих ситуацій, пов’язаних з наявністю або характеристиками дефектів.

Якщо сканована поверхня в нормі має координати Z, що знаходяться в певному діапазоні (наприклад, близько нуля або вище), то точки з координатами Zзначно меншими за нуль можуть вказувати на заглиблення, вм’ятини або інші види дефектів на поверхні. Ці заглиблення можуть бути результатом механічних пошкоджень, корозії, ерозії або виробничих дефектів. У випадку тріщини, особливо якщо вона має певну ширину і глибину, точки всередині розкритої тріщини можуть мати z-координати значно менші, ніж основна поверхня навколо неї. Це дозволяє ідентифікувати не лише наявність тріщини, але й оцінити її глибину або ширину.

Корозія може призвести до втрати матеріалу з поверхні, створюючи ділянки зі зниженими координатами Z. Особливо це стосується пітингової корозії, яка утворює глибокі точкові ураження. Значний знос або деформація поверхні може призвести до локальних областей зі зниженими координатами Z порівняно з номінальною формою.

Якщо на поверхні є пори або сторонні включення, які заглиблені відносно основної поверхні, їх скановані точки також можуть мати менші координати Z.

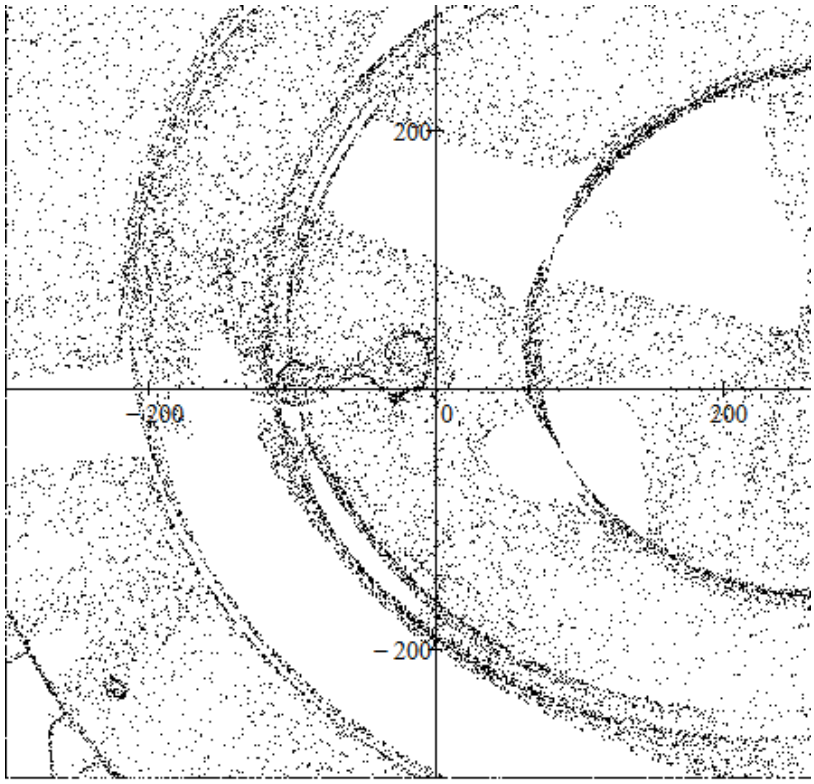


Рис. 4.7 – Результати четвертого етапу обробки хмари точок: *n*=19550

Важливо зазначити, що значення координат Z залежить від обраної системи координат під час сканування та обробки даних. Тому порівняння з нулем має розглядатися у контексті загальної геометрії об’єкта та налаштувань сканера.

На п’ятому етапі обираємо точки, координата N_Z нормалі яких буде менше експериментально визначеного порогового значення $N_Z < 0.5$ (рис. 4.8). Координата N_Z нормалі точки з хмари точок сканованої поверхні буде менше 0.5 (для одиничного вектора нормалі), якщо локальна поверхня в цій точці значно відхиляється від горизонтальної орієнтації, де вісь z спрямована вгору. На цьому етапі можливе виявлення наступних особливостей сканованої поверхні та дефектів:

1. Похилі або вертикальні ділянки поверхні. Значення $N_Z < 0.5$ означає, що вектор нормалі має значні складові вздовж осей x та/або y. Геометрично це відповідає ділянкам поверхні, які не є горизонтальними, а мають значний нахил або навіть є вертикальними. Самі по собі похилі або вертикальні ділянки не є дефектами, якщо вони відповідають геометрії об’єкта.

2. Край дефекту. У багатьох випадках краї дефектів, таких як тріщини, сколи, вм’ятини або виступи, характеризуються різкою зміною орієнтації поверхні. Точки, розташовані на краю такого дефекту, часто матимуть нормалі, значно відхилені від нормалей основної, поверхні без дефектів. Якщо основна поверхня є переважно горизонтальною, з нормаллю, близькою до $[N_X, N_Y, N_Z] = [0,0,1]$, то на краях дефекту N_Z може бути значно меншим за 0.5.

3. Поверхня дефекту, орієнтована під кутом. Сам дефект може мати поверхню, орієнтовану під певним кутом до основної поверхні. Наприклад, скол або тріщина може мати фаски або нерівні краї, де нормалі будуть мати значні N_X та N_Y складові, що призведе до малої N_Z складової.

4. Нерівності та шорсткість. На шорсткій або нерівній поверхні локальні нормалі можуть сильно коливатися. Якщо загальна тенденція поверхні є горизонтальною, то окремі мікронерівності можуть мати нормалі зі значно меншою N_Z складовою. Однак у дефектоскопії зазвичай розглядаються більш макроскопічні відхилення.

5. Виступи або нарости. Подібно до заглиблень, виступи або нарости на поверхні також можуть мати краї та власні поверхні, орієнтовані під кутом до основної поверхні, що призведе до меншого значення N_Z.

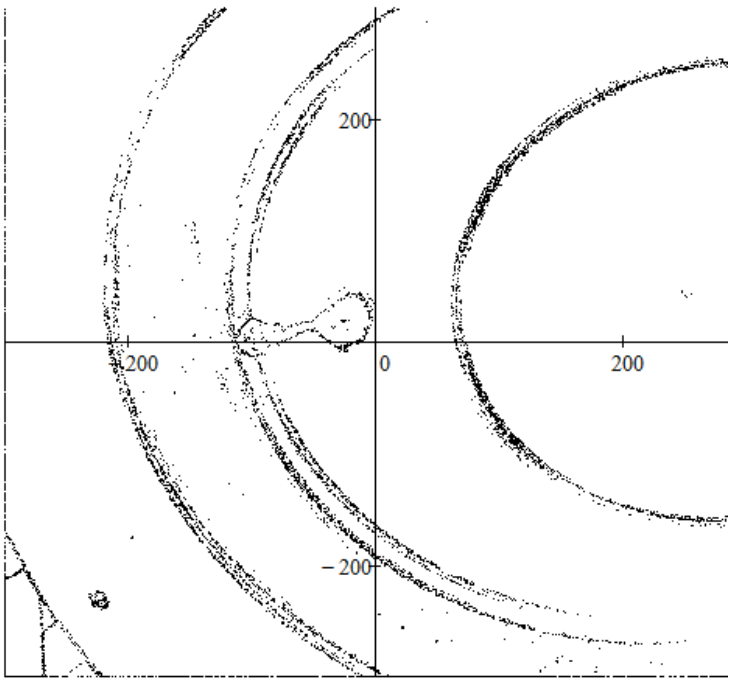


Рис. 4.8 – Результати п’ятого етапу обробки хмари точок: *n*=10170

Для остаточної ідентифікації дефектів потрібно виконати порівняння опрацьованої хмари точок сканованої поверхні з референтною поверхнею (номінальною моделлю або з сусідніми ділянками без дефектів) для виділення ділянок поверхні, що відповідають геометрії об’єкта.

Порівняння результатів поточного сканування з попередніми результатами дозволяє відстежувати зміни, які відбулися з часом, що особливо важливо для виявлення прогресуючих дефектів, таких як корозія, тріщини, деформації або знос.

Регулярне порівняння може виявити дефекти на ранніх стадіях їх розвитку, коли вони ще не є критичними і їх усунення є менш витратним. Порівняння допомагає виявити нові пошкодження, які могли виникнути внаслідок експлуатації, зовнішніх впливів або інших факторів.

4.6. Структура автоматизованої системи сканування внутрішньої поверхні автоклава

На основі проведених досліджень розроблена автоматизована система сканування внутрішньої поверхні автоклавів (рис. 4.9), що ґрунтується на ефективних методах сканування, які враховують складність форми та можливі перешкоди, а також на розв’язанні зворотної задачі кінематики [24, 29, 52] для керування сферичним паралельним механізмом, що забезпечує рух діагностичного інструменту по базових траєкторіях, завдяки чому система демонструє гнучкість та можливість адаптації до різних типів і розмірів промислових автоклавів.

Автоматизована система сканування поверхні автоклаву зі сферичним паралельним механізмом функціонує під керуванням операторського інтерфейсу (HMI), який забезпечує введення параметрів сканування, відображення процесу, візуалізацію результатів та генерацію звітів. Центральний контролер здійснює загальне керування системою, координує роботу всіх компонентів та зберігає конфігураційні дані.

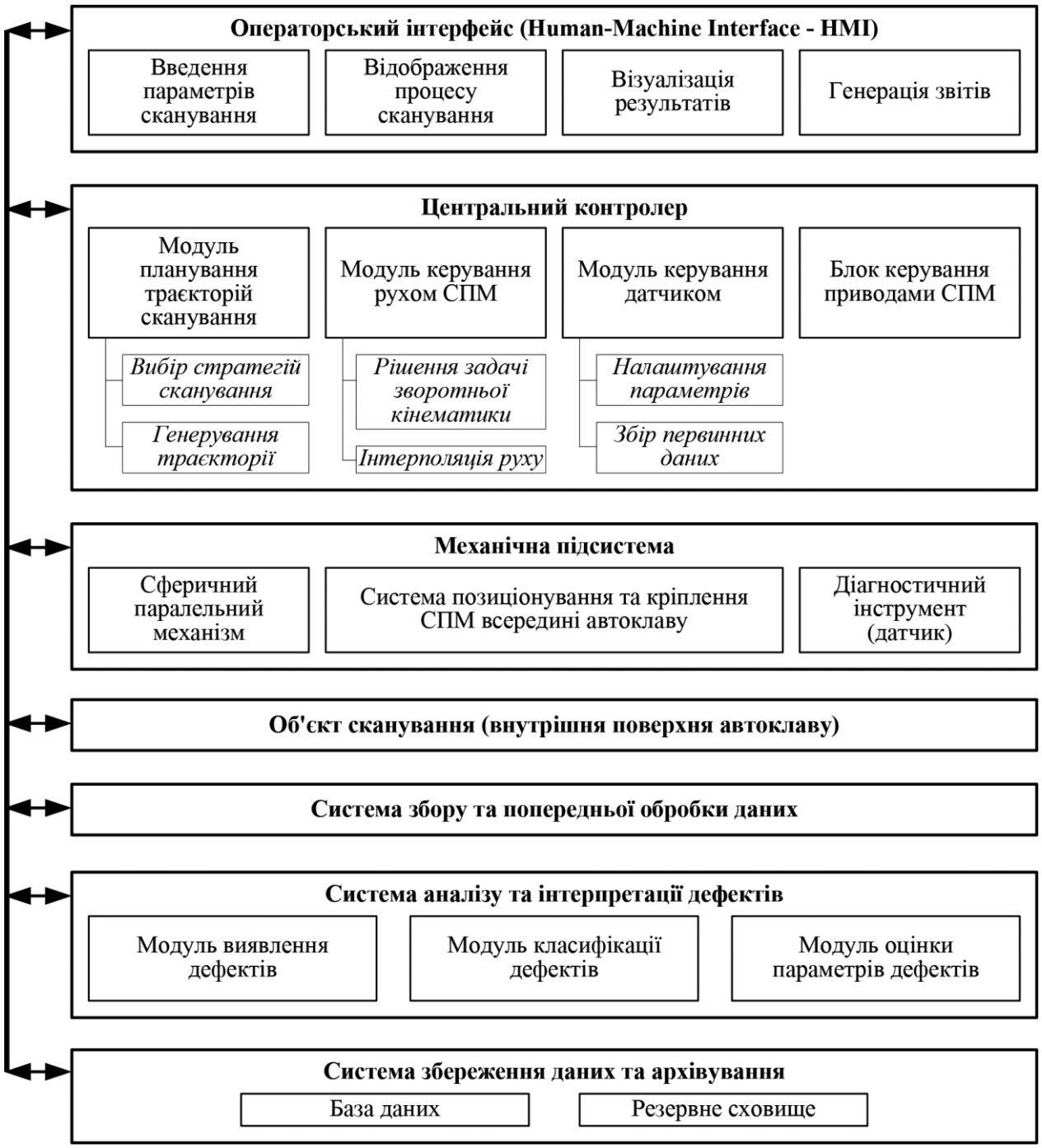


Рис. 4.9 – Структура автоматизованої системи діагностики поверхонь автоклавів з використанням сферичного паралельного механізму

Керування рухом діагностичного інструменту забезпечується модулем планування траєкторії, який обирає стратегію сканування, генерує точки траєкторії та оптимізує рух, а також модулем керування рухом СПМ, що вирішує задачу зворотньої кінематики, здійснює інтерполяцію руху та керує приводами [26, 29].

Робота діагностичного датчика, у якості якого може застосовуватися візуальна камера, ультразвуковий або лазерний сканер, контролюється окремим модулем керування датчиком, яка налаштовує параметри, запускає та зупиняє збір первинних даних. Приведення в рух СПМ, що складається з механічної платформи, приводів, ланок та шарнірів, здійснює блок керування приводами СПМ.

Отримані від датчика первинні дані надходять до системи збору та попередньої обробки даних, де відбувається фільтрація, калібрування та сегментація. Подальший аналіз та інтерпретація даних здійснюється для виявлення дефектів, їх класифікації та оцінки параметрів. На завершальному етапі система збереження даних та архівування забезпечує постійне та довгострокове зберігання сирих, оброблених і проаналізованих даних сканування, а також надає інструменти для пошуку, перегляду та експорту даних для подальшого аналізу та звітності про результати сканування внутрішньої поверхні автоклаву, яка є об’єктом дослідження.

4.7. Висновки до розділу 4

Проведене дослідження підтвердило перспективність використання СПМ в структурі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів завдяки його високій маневреності та здатності орієнтувати діагностичний інструмент у широкому діапазоні кутів, що є важливим для сканування складних геометричних форм. Геометрія внутрішньої поверхні є визначальним фактором при виборі стратегії сканування. Задача діагностики (виявлення загальних дефектів, пошук конкретних типів дефектів, оцінка розмірів пошкоджень) визначає необхідну щільність сканування та складність траєкторії. У складних сценаріях діагностики може використовуватися комбінація розглянутих базових траєкторій або їх адаптація в режимі реального часу на основі даних, що надходять від датчика. При інтеграції СПМ в автоматизовані системи діагностики рухи СПМ повинні бути запрограмовані заздалегідь або генеруватися в режимі реального часу на основі

даних від сенсорів. Для реалізації траєкторій сканування внутрішньої поверхні автоклава, обходу перешкод або фокусування на певних областях, необхідно постійно розраховувати зворотню задачу кінематики для визначення потрібних положень приводів на кожному кроці руху. Розроблена автоматизована система діагностики з використанням СПМ демонструє гнучкість та можливість адаптації до різних типів і розмірів автоклавів.

Аналіз сучасних методів сканування показав перспективність використання хмар точок для детального відображення поверхні автоклавів. Застосування фотограмметрії як методу отримання хмар точок є економічно вигідним та дозволяє сканувати об’єкти великих розмірів з високою деталізацією. Розроблений алгоритм обробки хмари точок, що включає етапи фільтрації на основі аналізу координат нормалей та координат точок, демонструє можливість ефективного виділення потенційних ділянок з дефектами.

Видалення точок з нульовими та близькими до одиниці значеннями компоненти N_Z нормалей дозволяє відфільтрувати ізольовані, планарні та горизонтальні ділянки, зосереджуючи увагу на геометричних аномаліях. Виділення точок з від’ємними значеннями координат Z є індикатором заглиблень та інших дефектів, пов’язаних з втратою матеріалу або зміною геометрії поверхні. Аналіз нормалей з малою компонентою N_Z дозволяє виявляти краї дефектів та похилі ділянки.

Проте для ефективного виявлення та аналізу дефектів необхідно враховувати не лише значення окремої компоненти нормалі, але й комплексний аналіз розподілу нормалей на сканованій поверхні, а також порівняння з очікуваними нормалями для поверхні без дефектів. Різкі зміни та аномальні значення нормалей часто є ключовими ознаками наявності дефектів.

Запропонований підхід до обробки хмар точок є важливим кроком до автоматизації процесу виявлення дефектів на поверхні автоклавів. Для остаточної ідентифікації дефектів необхідним є порівняння опрацьованої хмари точок з референтною моделлю або результатами попередніх сканувань, що дозволить виділити значущі відхилення та відстежувати динаміку розвитку

дефектів з часом, підвищуючи ефективність контролю якості, економічність обслуговування та безпеку експлуатації автоклавів.

Розроблено структуру автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів з використанням СПМ, яка включає операторський інтерфейс, центральний контролер, модулі планування траєкторій та керування рухом, систему збору та обробки даних.

Отримані результати підтверджують перспективність використання сферичного паралельного механізму в автоматизованих системах діагностики та демонструють гнучкість і можливість адаптації системи до різних типів і розмірів промислових об'єктів.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальне науково-технічне завдання підвищення точності ідентифікації параметрів руху елементів каркасних конструкцій на основі сферичних паралельних механізмів шляхом розробки математичних моделей, методів фільтрації та програмно-апаратних засобів керування. Основні наукові та практичні результати роботи полягають у наступному:

1. Проведено аналіз МПС, їх типів, принципів побудови, методів математичного моделювання, технічних засобів реалізації систем керування та методів ідентифікації параметрів руху. Обґрунтовано доцільність використання методу, що поєднує дані, отримані за допомогою вимірювальних засобів, з математичними методами ідентифікації та підвищує достовірність оцінювання параметрів руху механічної системи порівняно з використанням окремих джерел інформації.

2. Розроблено комплексну математичну модель СПМ, що охоплює: геометричний опис; модель динаміки параметрів руху на основі розв'язання зворотної задачі кінематики з використанням законів аналітичної геометрії; модель орієнтації платформи на основі матричних перетворень; інкрементальний підхід до визначення кутового положення; врахування геометричних та кінематичних обмежень; метод вибору єдиної допустимої конфігурації на основі багатокрокової фільтрації; визначення робочої області. Розрахунковий експеримент і перевірка на основі даних MPU6050 у статичних режимах підтвердила високу відповідність моделі реальним даним: середньоквадратична похибка не перевищує 0,35°, а відносна похибка — 2,11 %, що свідчить про адекватність моделі та її придатність для систем керування.

3. Розроблено метод ідентифікації параметрів руху елементів СПМ на основі алгоритму комплементарного фільтра, адаптованого для систем керування з обмеженими обчислювальними ресурсами, що поєднує дані гіроскопа та акселерометра для оцінки кута нахилу рухомої платформи, зменшуючи вплив дрейфу гіроскопа та шумів акселерометра. Експериментальна перевірка на СПМ

підтвердила працездатність та ефективність запропонованого підходу. Динамічна похибка запропонованого комплементарного фільтра не перевищує 0,8° при коефіцієнті довіри в діапазоні 0,9–0,98. При цьому необхідний обсяг пам’яті становить приблизно 100 байт, а час обчислення не перевищує 1 мкс.

4. Розроблено програмно-апаратну платформу керування СПМ у режимі реального часу, що забезпечує інтелектуальну обробку відеоданих та відстеження положення об'єкта керування за допомогою камери. Система реалізована у вигляді дворівневої архітектури з клієнтським графічним інтерфейсом та серверною частиною для обчислення задачі оберненої кінематики і формування керуючих сигналів. Розроблено програмний модуль визначення положення об'єкта та засоби візуалізації роботи системи у дво- та тривимірному просторі.

5. Доведено, що розроблений метод безконтактного дистанційного керування СПМ на основі інтелектуальної обробки відеоданих забезпечує ефективне керування сферичними паралельними механізмами в режимі реального часу без використання контактних вимірювальних засобів.

6. Досліджено практичні аспекти застосування СПМ в складі автоматизованої системи діагностики внутрішніх поверхонь промислових автоклавів. Запропоновано використання алгоритму обробки хмари точок на основі аналізу координат нормалей та просторових координат точок, що дозволяє ефективно виділяти потенційні ділянки з дефектами. Ефективність запропонованого методу підтверджено скороченням отриманої фотограмметричним методом початкової хмари точок в 10,15 разів.

7. Удосконалено структуру автоматизованої системи діагностики внутрішніх поверхонь автоклавів, яка включає: операторський інтерфейс, центральний контролер, модулі планування траєкторій та керування рухом, систему збору та обробки даних.

Отримані результати підтверджують перспективність використання сферичних паралельних механізмів у складі автоматизованих систем діагностики та демонструють можливість адаптації розроблених методів до різних типів промислового обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zhang, Z., Qizhi, M., & Zhiwei, C. et al. (2025). Machine Learning Applications in Parallel Robots: A Brief Review. *Machines*, 13(7), 565. DOI: 10.3390/machines13070565.

2. Russo M., Zhang D., & Liu X.-J. et al. (2024). A review of parallel kinematic machine tools: Design, modeling, and applications. *International Journal of Machine Tools and Manufacture*. DOI: 10.1016/j.ijmachtools.2024.104118.

3. Kumar S., Wöhrle H., & de Gea Fernández J. et al. (2020). A survey on modularity and distributivity in series-parallel hybrid robots. *Mechatronics*, 68, 102367. DOI: 10.1016/j.mechatronics.2020.102367.

4. Deabs A., Gomaa, F. R., Khader, K. M. (2021). Parallel Robot – Review Article. *Journal of Engineering Science and Technology Review*, 14(6). DOI: 10.25103/jestr.146.02.

5. Wang L., Zhang J. W., Zhang D. (2025). A Review on Reconfigurable Parallel Mechanisms: Design, Analysis and Challenge. *Engineering*. Vol. 47(4). 108 - 124. DOI: 10.1016/j.eng.2024.09.022.

6. Chablat D., Michel G., & Bordure P. et al. (2021). Workspace analysis in the design parameter space of a 2-DOF spherical parallel mechanism for a prescribed workspace: Application to the otologic surgery. *Mechanism and Machine Theory*, 157, 104224. DOI: 10.1016/j.mechmachtheory.2020.104224

7. Chen, X., Xin, C., & Zhang, Z. et al. (2024). Design Principles and Kinematic Analysis of a Novel Spherical 2-DOF Parallel Mechanism. *Mechanical Sciences*, *15*, 473–486. DOI: 10.5194/ms-15-473-2024

8. Tursynbek I., Niyetkaliye A., Shintemirov A. (2019). Computation of Unique Kinematic Solutions of a Spherical Parallel Manipulator with Coaxial Input Shafts. *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 1524–1531. DOI: 10.1109/COASE.2019.8843090.

9. Taunyazov T., Rubagotti M., Shintemirov A. (2018). Constrained Orientation Control of a Spherical Parallel Manipulator via Online Convex

Optimization. *IEEE/ASME Transactions on Mechatronics*, 23(1), 252–261. DOI: 10.1109/TMECH.2017.2774245.

10. Saafi H., Laribi M.A., Zeghloul S. Forward Kinematic Model Resolution of a Special Spherical Parallel Manipulator: *Comparison and Real-Time Validation*. Robotics. 2020. Vol. 9(3). 62.

11. Saik N., Tadakuma K., & Watanabe M. et al. (2021). 2-DOF Spherical Parallel Mechanism Capable of Biaxial Swing Motion with Active Arc Sliders. *IEEE Robotics and Automation Letters*, *6*(3), 4680–4687. DOI: 10.1109/LRA.2021.3064187

12. Lê A., Chablat D., Rance G., Rouillier F. On the Certification of the Kinematics of 3-DOF Spherical Parallel Manipulators. *Maple Transactions*. 2023. Vol. 3(2). Article 15660.

13. Ru X., Gu N., & Shang H. et al. (2022). MEMS Inertial Sensor Calibration Technology: Current Status and Future Trends. *Micromachines*, 13(6). 879. DOI: 10.3390/mi13060879.

14. Примушко А. М., Рижков Л. М. Дослідження комплементарного фільтра на МЕМС-вимірювачах. *Інформаційні системи, механіка та керування*. 2019. Вип. 20. С. 47–53.

15. Mahony R., Hamel T., Pflimlin J.-M. (2008). Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Transactions on Automatic Control*, 53(5), 1203–1218. DOI: 10.1109/TAC.2008.923738.

16. Madgwick S. O. H., Harrison A. J. L., Vaidyanathan R. (2011). Estimation of IMU and MARG orientation using a gradient descent algorithm. *2011 IEEE International Conference on Rehabilitation Robotics*, 1–7. DOI: 10.1109/ICORR.2011.5975346.

17. Шмана К. С. Використання фільтрів Калмана та Маджвіка для обробки показань гіроскопу та акселерометра. *Новітні технології у науковій діяльності і навчальному процесі : матеріали тез доп. Всеукр. наук.-практ. конф. студентів, аспірантів та молодих учених*. Чернігів, 2019. С. 85–87.

18. Gao W., Li J., Zhou G., Li Q. (2015). Adaptive Kalman Filtering with Recursive Noise Estimator for Integrated SINS/DVL Systems. *Journal of Navigation*, 68(1), 142–161. DOI: 10.1017/S0373463314000484.

19. Arents J., Greitans M. (2022). Smart Industrial Robot Control Trends, Challenges and Opportunities within Manufacturing. *Applied Sciences*, 12(2), 937. DOI: 10.3390/app12020937.

20. Shahria M. T., Sunny M. S. H., & Zarif M. I. I. et al. (2022). A Comprehensive Review of Vision-Based Robotic Applications: *Current State, Components, Approaches, Barriers, and Potential Solutions. Robotics*, 11(6). 139. DOI: 10.3390/robotics11060139.

21. Shapiro L. G., Stockman G. C. (2001). Computer Vision. Upper Saddle River, NJ: Prentice Hall.

22. docs.opencv.org. OpenCV-Python Tutorials [Software documentation]. Retrieved from https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html.

23. Derkachenko A., Polyvoda O., Lebedenko Y., Kalinina K. (2023). Research of Control Methods of a Spherical Parallel Mechanism Using Intelligent Data Processing. *2023 IEEE 5th International Conference on Modern Electrical and Energy System (MEES)*, 1–5. DOI: 10.1109/MEES61502.2023.10402530.

24. Деркаченко А. О., Поливода О. В. Автоматизована система діагностики внутрішніх поверхонь промислових автоклавів з використанням сферичного паралельного механізму. *Вісник херсонського національного технічного університету*. 2025. № 2 (93), Ч. 1. С. 48-55

25. Деркаченко А. О., Поливода О. В. Аналіз засобів вимірювання параметрів руху елементів каркасних конструкцій. *Матеріали X Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління присвяченої Дню ракетно-космічної галузі України*. Херсон – Хмельницький, 2023. С. 15–18.

26. Деркаченко А. О., Поливода О. В., Русанов С. А., Лебеденко Ю. О. Керування приводами сферичного паралельного механізму з використанням мови програмування MAXSCRIPT. *Матеріали V всеукраїнської науково-технічної конференції «Інформаційні технології та програмування»*. Херсон, 27 грудня 2023 р. Хмельницький, 2023. С. 17–21.

27. Деркаченко А. О., Поливода О. В. Математична модель руху елементів каркасних конструкцій. *Матеріали XI Всеукраїнської науково-*

практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління присвяченої Дню ракетно-космічної галузі України : Збірник наукових праць / за ред. Г. В. Рудакової та ін. Херсон-Хмельницький, 2024. С. 41–45.

28. Деркаченко Т. О., Поливода О. В., Деркаченко А. О. Розробка комплексної математичної моделі системи просторового позиціонування сонячного трекера як механізму паралельної структури. *Прикладні питання математичного моделювання*. 2025. Т. 8, Ч. 2. С. 93–101.

29. Деркаченко А. О., Поливода О. В. Розробка алгоритму визначення положення платформи сферичного паралельного механізму в 3D-просторі та його програмна реалізація. *Вісник Херсонського національного технічного університету*. 2025. Т. 1, № 3 (94). С. 79–87. DOI: 10.35546/kntu2078-4481.2025.3.1.9

30. Деркаченко А. О., Поливода О. В. Використання комплементарного фільтра для ідентифікації параметрів руху елементів сферичного паралельного механізма. *Проблеми інформатизації та управління*. 2025. № 1 (81). С. 23–29. DOI: 10.18372/2073-4751.81.20125

31. Поливода О. В., Деркаченко А. О., Поливода В. В. Автоматизоване розпізнавання дефектів поверхні автоклавів з використанням аналізу хмар точок. *Прикладні питання математичного моделювання*. 2025. Т. 8, Ч. 1. С. 147–155. DOI: 10.32782/mathematical-modelling/2025-8-1-14

32. Дмитрієв Д. О., Русанов С. А., Деркаченко А. О., Лебеденко Ю. О., Поливода О. В. Комп'ютерна програма «ToolsSM» : свідоцтво про реєстрацію авторського права на твір № 111283 від 24.01.2022 р.

33. Деркаченко А. О., Поливода О. В. Комп'ютерна програма «СПМ API Server» : свідоцтво про реєстрацію авторського права на твір № 141922.

34. Lebedenko Y., Polyvoda O., & Derkachenko A. et al. (2022). Research of Control Systems for Robotic Spatial Planning Platforms. *2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES)*, 1–4. DOI: 10.1109/MEES58014.2022.10005765.

35. Plevris V., Ahmad A. (2025). Deriving analytical solutions using symbolic matrix structural analysis: Part 2 – Plane trusses. Heliyon, 11(4), e42372. DOI: 10.1016/j.heliyon.2025.e42372.

36. Megson T. H. G. (2014). Structural and Stress Analysis (4th ed.). Oxford: Butterworth-Heinemann.

37. Piątkowski M., Sempruch J. (2021). Robustness Analysis and Important Element Evaluation Method of Truss Structures. *Buildings*, 11(10), 436. DOI: 10.3390/buildings11100436.

38. Основи мехатроніки / О. М. Артюх, О. В. Дударенко, В. В. Кузьмін та ін. Запоріжжя : НУ «Запорізька політехніка», 2021. 372 с.

39. Стрілець О. Р. Теорія механізмів і машин: Конспект лекцій. Рівне : НУВГП, 2023. 229 с.

40. Bai S. Kinematics of Computationally Efficient Mechanisms. Mechanism and Machine Theory. 2024. Vol. 203. DOI: 10.1016/j.mechmachtheory.2024.105782.

41. Chen Z., Chen X., & Gao M. et al. (2022). Motion Characteristics Analysis of a Novel Spherical Two-degree-of-freedom Parallel Mechanism. *Chinese Journal of Mechanical Engineering*, *35*, 29. DOI: 10.1186/s10033-022-00702-7

42. Кузнєцов Ю. М. Компоновка верстатів з механізмами паралельної структури : монографія / Ю. М. Кузнєцов, Д. О. Дмитрієв, Г. Ю. Діневич ; за ред. Ю. М. Кузнєцова. Херсон : ПП Вишемирський В. С., 2009. 456 с.

43. Jin X.-D., Fang Y.-F., & Gou S. et al. (2018). Structural Synthesis of Parallel Mechanisms with High Rotational Capability. Chinese Journal of Mechanical Engineering, 31. DOI: 10.1186/s10033-018-0260-3.

44. Simas H., Di Gregorio R., Simoni R. (2023). Instantaneous Kinematics and Free-from-Singularity Workspace of 3-XXRRU Parallel Manipulators. Robotics, Vol. 12(5), 138. DOI: 10.3390/robotics12050138.

45. Nigatu H., Kim D. (2023). Workspace optimization of 1T2R parallel manipulators with a dimensionally homogeneous constraint-embedded Jacobian. Mechanism and Machine Theory. DOI: 10.1016/j.mechmachtheory.2023.105391.

46. Mueller A. (2024). A Constraint Embedding Approach for Dynamics Modeling of Parallel Kinematic Manipulators with Hybrid Limbs. Robotics and Autonomous Systems. DOI: 10.1016/j.robot.2022.104187.

47. Mueller A. (2024). Dynamics of Parallel Manipulators with Hybrid Complex Limbs – Modular Modeling and Parallel Computing. Mechanism and Machine Theory. DOI: 10.1016/j.mechmachtheory.2021.104549.

48. Valles M., Araujo-Gomezet P. & Mata V. et al. (2024). Mechatronic Design, Experimental Setup and Control Architecture Design of a Novel 4 DoF Parallel Manipulator. arXiv:2401.08192. DOI: 10.48550/arXiv.2401.08192.

49. Petrović G. R., Mattila J. (2021). Mathematical modelling and virtual decomposition control of heavy-duty parallel–serial hydraulic manipulators. Mechanism and Machine Theory. DOI: 10.1016/j.mechmachtheory.2021.104680.

50. Wu J., Ye H., & Yu G. et al. (2022). A novel dynamic evaluation method and its application to a 4-DOF parallel manipulator. Mechanism and Machine Theory, 168, 104627. DOI: 10.1016/j.mechmachtheory.2021.104627.

51. Wang L., Zhang J. W., Zhang D. (2025). A Review on Reconfigurable Parallel Mechanisms: Design, Analysis and Challenge. Engineering. 47(4). DOI: 10.1016/j.eng.2024.09.022.

52. Wu L., Tron R. (2024). Inverse Kinematics with Vision-Based Constraints. arXiv:2406.10682. DOI: 10.48550/arXiv.2406.10682.

53. Лебеденко Ю.О., Омельчук А.А., Саф'яник О.О. Інформаційно-вимірювальна підсистема багатоприводної каркасної установки з механізмами паралельної структури. Вісник херсонського національного технічного університету. 2017. № 3 (62). Ч. 1. с. 317–322.

54. Ультразвукові далекоміри. Переваги і недоліки в порівнянні з лазерними рулетками. URL: https://mpr-kip.com/ua/a292496-ultrazvukovye-dalnomery-dostoinstva.html (дата звернення: 09.06.2026)

55. Основні види лазерних далекомірів. URL: https://simvolt.ua/osnovni-vidi-lazernikh-dalekomiriv/ (дата звернення: 09.06.2026).

56. Чмельов В. О. Радіолокаційні системи : навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 172 «Електронні комунікації та радіотехніка». Київ : КПІ ім. Ігоря Сікорського, 2023. 60 с.

57. Electronic Components Datasheet Search. (2012). MPU-6050 Datasheet [Datasheet]. Retrieved from https://www.alldatasheet.com/datasheet-pdf/pdf/517744/ETC1/MPU-6050.html.

58. Akbaba C. E., Tanrikulu M. Y. (2023). MEMS capacitive accelerometer: A review. Artıbilim: Adana Alparslan Türkeş Bilim ve Teknoloji Üniversitesi Fen Bilimleri Dergisi, 6(2), 41–58. DOI: 10.55198/artibilimfen.1386846.

59. Arduino Documentation. Learn Basic knowledge about principles and techniques behind the Arduino ecosystem. [Documentation]. Retrieved from https://docs.arduino.cc/learn/.

60. Chenchireddy K., Dora R., Mulla G.B., Jegathesan V., Sydu S.A. (2024). Development of robotic arm control using Arduino controller. IAES International Journal of Robotics and Automation (IJRA), 13(3), 264–271. DOI: 10.11591/ijra.v13i3.pp264-271.

61. Hercog D., Lerher T., Truntič M., Težak O. (2023). Design and Implementation of ESP32-Based IoT Devices. Sensors, 23(15), 6739. DOI: 10.3390/s23156739.

62. Tsai S.-E., Yang S.-M., Sun W.-C. (2025). The Study on Real-Time RRT-Based Path Planning for UAVs Using a STM32 Microcontroller. Electronics, 14(24), 4901. DOI: 10.3390/electronics14244901.

63. Greco D., Fasihiany M., Ranjbar A.V., Masulli F., Rovetta S., Cabri A. (2024). Computer Vision Algorithms on a Raspberry Pi 4 for Automated Depalletizing. Algorithms, 17(8), 363. DOI: 10.3390/a17080363.

64. Hakani R., Rawat A. (2024). Edge Computing-Driven Real-Time Drone Detection Using YOLOv9 and NVIDIA Jetson Nano. Drones, 8(11), 680. DOI: 10.3390/drones8110680.

65. Kutia M., Laktionov I., Kyselov V., Hlushko A. (2024). Principles and Methods of Servomotor Control: Comparative Analysis and Applications. Applied Sciences, 14(6), 2579. DOI: 10.3390/app14062579.

66. Mueller, J. P. (2018). *Beginning programming with Python for dummies* (2nd ed.). John Wiley & Sons.

67. pyserial.readthedocs.io. Welcome to pySerial's documentation [Software documentation]. Retrieved from https://pyserial.readthedocs.io/en/latest/.

68. Marcon P., Zezulka F., Vesely I., Szabo Z., Roubal Z., Sajdl O., Jambor B., Dohnal P. (2023). Assessing Industrial Communication Protocols to Bridge the Gap between Machine Tools and Software Monitoring. Sensors, 23(12), 5694. DOI: 10.3390/s23125694.

69. Wang Z., Li X., Zhang H., Chen Y. (2024). The Design and Real-Time Optimization of an EtherCAT Master for Multi-Axis Motion Control. Electronics, 13(15), 3101. DOI: 10.3390/electronics13153101.

70. Vu L.A.T., Bi Z., Mueller D., Younis N. (2023). Modular Self-Configurable Robots—The State of the Art. Actuators, 12(9), 361. DOI: 10.3390/act12090361.

71. Zhang X., Shi R., & Zhu Z. et al. (2023). Adaptive nonsingular fixed-time sliding mode control for manipulator systems' trajectory tracking. Complex & Intelligent Systems, 9(2), 1605–1616. DOI: 10.1007/s40747-022-00864-w.

72. Mediapipe Face Mesh. (2024). [Face mesh detection tool]. Retrieved from https://developers.google.com/edge/mediapipe/solutions/vision/face_landmarker.

73. Mediapipe Holistic. (2024). [Holistic body detection tool]. Retrieved from https://developers.google.com/edge/mediapipe/solutions/vision/holistic_landmarker

74. Andel J., Šimák V., & Kanálikova A. el al. (2022). GNSS Based Low-Cost Magnetometer Calibration. Sensors, 22(21), 8447. DOI: 10.3390/s22218447.

75. Ito T., Yoneyama H., & Akiyama Y. et al. (2023). Sensing Algorithm to Estimate Slight Displacement and Posture Change of Target from Monocular Images. Sensors, 23(2), 851. DOI: 10.3390/s23020851.

76. Servières M., Renaudin V., & Dupuis A. et al. (2021). Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking. Journal of Sensors, 2021(1), DOI: 10.1155/2021/2054828.

77. Ribeiro-Gomes J., Gaspar J., Bernardino A. (2023). Event-Based Feature Tracking in VIO. Frontiers in Robotics and AI. 10. DOI: 10.3389/frobt.2023.994488.

78. Zhu J., Li H., Zhang T. (2024). Camera, LiDAR, and IMU Based Multi-Sensor Fusion SLAM: A Survey. Tsinghua Science and Technology, 29(2), 415–429. DOI: 10.26599/TST.2023.9010010.

79. Khairallah M., et al. (2023). Flow-Based Event VIO. Proceedings of the International Conference on Pattern Recognition (ICPR). DOI: 10.5220/0011660400003417.

80. Yang Y., Geneva P., Zuo X., Huang G. (2023). Online Self-Calibration for Visual-Inertial Navigation: Models, Analysis, and Degeneracy. IEEE Transactions on Robotics. DOI: 10.1109/TRO.2023.3275878.

81. Ušinskis V., Nowicki M., & Dzedzickis A. et al. (2025). Sensor-Fusion Based Navigation for Autonomous Mobile Robot. Sensors, 25(4), 1248. DOI: 10.3390/s25041248

82. Li X., Lui C., Yan X. (2025). Deepline-VIO. Sensors, 25(16), 5029. DOI: 10.3390/s25165029.

83. Kühne J., Magno M., Benini L. (2025). Low-Latency VIO. IEEE Sensors Journal. DOI: 10.1109/JSEN.2024.3406948.

84. von Stumberg L., Cremers D. (2022). DM-VIO. arXiv:2201.04114. DOI: 10.48550/arXiv.2201.04114.

85. Qin T., Li P., Shen S. (2023). VINS-Mono / VINS-Fusion. arXiv:1708.03852. DOI: 10.48550/arXiv.1708.03852.

86. Dhédin V., Li H., & Khorshidi S. et al. (2022). Visual–Inertial–Kinematic Fusion. arXiv:2210.02127. DOI: 10.48550/arXiv.2210.02127.

87. Лебеденко Ю. О., Рудакова Г. В., Тоуфак Е. Р. Моделювання параметрів руху елементів технологічних установок каркасної конструкції. Вісник херсонського національного технічного університету. 2017. № 3 (62). Ч.2. С. 128-132.

88. Walia, K., Navaraj, W., & Breedon, P. (2025). Generatively Optimised Compact 3-DoF Spherical Parallel Manipulator with Tightly Integrated Actuators. *Frontiers in Mechanical Engineering*, 11. DOI: 10.3389/fmech.2025.1629405

89. Ye W., Li Q., Chai X. (2022). Type Synthesis of 4-DOF Non-Overconstrained Parallel Mechanisms with Symmetrical Structures. Machines, 10(12), 1123. DOI: 10.3390/machines10121123.

90. Дімітрова-Бурлаєнко С. Д. Розв'язання задач аналітичної геометрії векторним методом : навч.-метод. посібник / С. Д. Дімітрова-Бурлаєнко, В. М. Бурлаєнко, Н. П. Гиря. 2-ге вид., випр. і доп. Харків : НТУ "ХПІ", 2020. 50 с.

91. Wang K. (2021). A theoretical analysis method of spatial analytic geometry and mathematics under digital twins. Advances in Civil Engineering, 2021, 8910274. DOI: 10.1155/2021/8910274.

92. Georgiev S.G., Zennir K., Boukarou A. (2022). Multiplicative analytic geometry. Taylor & Francis.

93. Magnetic interference testing method for an electric fixed-wing UAS. J. Unmanned Veh. Sys., 2019. DOI: 10.1139/juvs-2018-0006

94. Kadam P., Fang G., Zou J. J. (2024). Object Tracking Using Computer Vision: A Review. Computers, 13(6), 136. DOI: 10.3390/computers13060136.

95. w3schools.com. Python math Module [Tutorial]. Retrieved from https://www.w3schools.com/python/module_math.asp.

96. NumPy. NumPy: Fundamental package for scientific computing with Python [Software library]. Retrieved from https://numpy.org/.

97. dlib.net. Dlib documentation [Software documentation]. Retrieved from http://dlib.net/python/index.html.

98. Matplotlib. Matplotlib: Visualization with Python [Software library]. Retrieved from https://matplotlib.org/.

99. Russell S. J., Norvig P. (2018). Artificial Intelligence: A Modern Approach. Boston: Pearson.

100. Постанова Кабінету Міністрів України «Про затвердження технічного регламенту обладнання, що працює під тиском» : прийнята 16 січня 2019 р. № 27. Офіційний вісник України. 2019. № 9. С. 90.

101. Поливода О. В., Лебеденко Ю. О. Ідентифікація систем автоматичного управління акустико-емісійною діагностикою. Методи та прилади контролю якості. 2020. № 1 (44). С. 35–45.

102. Wang J., Zhang J., Xu Q. (2014). Research on 3D laser scanning technology based on point cloud data acquisition. 2014 International Conference on Audio, Language and Image Processing, 631–634. DOI: 10.1109/ICALIP.2014.7009871.

103. Дорожинський О. Л. Фотограмметрія та дистанційне зондування: актуальний стан і тенденції вдосконалення. Сучасні досягнення геодезичної науки та виробництва. 2011. Вип. II (22). С. 34–39.

104. Kvasov A., Penza V., et al. (2020). Introduction to Radar Scattering Application in Remote Sensing and Diagnostics: Review. Atmosphere, 11(5), 517. DOI: 10.3390/atmos11050517.

105. Orthey A., Chamzas C., Kavraki L.E. (2024). Sampling-Based Motion Planning: A Comparative Review. Annual Review of Control, Robotics, and Autonomous Systems, 7, 285–310. DOI: 10.1146/annurev-control-061623-094742.

106. Palmieri L., Koenig S., Arras K. O. (2016). RRT-Based Nonholonomic Motion Planning Using Any-Angle Path Biasing. 2016 IEEE International Conference on Robotics and Automation (ICRA), 2775–2781. DOI: 10.1109/ICRA.2016.7487439.

107. Revenko S. V., Omelchuk A. A., & Lebedenko Y. O. et al. (2018). Simulator of Motion of Wireframe Construction of Spatial Layout. 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), 187–190. DOI: 10.1109/MSNMC.2018.8576305.

108. Furukawa Y., Hernández C. (2013). Multi-View Stereo: A Tutorial. Foundations and Trends in Computer Graphics and Vision, 9(1–2), 1–148. DOI: 10.1561/0600000052.

109. Abdi H., Williams L.J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2, 433–459. DOI: 10.1002/wics.101.

110. Bishop C.M. Pattern recognition and machine learning. Springer, 2006. 778 p.

ДОДАТОК А

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 111283

Комп'ютерна програма «ToolsSM»

Автор(и): Дмитрієв Дмитро Олексійович, Русанов Сергій Аркадійович, Деркаченко Анатолій Олегович, Лебєгенко Юрій Олександрович, Поливода Оксана Валеріївна

Дата реєстрації 24 січня 2022 р.

Генеральний директор
Державного підприємства
«Український інститут
інтелектуальної власності»

Андрій КУДІН

УКРАЇНА

СВІДОЦТВО

про реєстрацію авторського права на твір

№ 141922

Комп'ютерна програма «SPM API Server» (Сервер централізованого керування модулями обчислення параметрів руху сферичного паралельного механізму (SPM))» («SPM API Server»)

(вид, назва твору)

Автор (співавтори) **Деркаченко Анатолій Олегович, Поливода Оксана Валеріївна**

(прізвище, ім'я, по батькові (за наявності), псевдонім (за наявності))

Авторські майнові права належать спільно **Деркаченко Анатолій Олегович, вул. Сонячна, 28, смт Антонівка, Херсонський р-н, Херсонська обл., 73486; Поливода Оксана Валеріївна, вул. Робоча, 201, кв. 19, м. Херсон, 73033**

(прізвище, ім'я, по батькові (за наявності) фізичної особи / найменування юридичної особи, адреса)

Дата реєстрації 27 січня 2026 р.

Директор Державної організації «Український національний офіс інтелектуальної власності та інновацій» **Олена ОРЛЮК**

УКРНОІВІ

ДОДАТОК Б

«ЗАТВЕРДЖУЮ»

ДИРЕКТОР ПРИВАТНОГО
ПІДПРИЄМСТВА "НАШ ПРОДУКТ"

Підпис ПІБ

"5" листопада 2025 р.

АКТ

про впровадження результатів дисертаційної роботи
Деркаченка Анатолія Олеговича

Методи ідентифікації параметрів руху елементів каркасних конструкцій

Комісія у складі:

Голова комісії:	Окуневич Ігор Мефодійович	головний інженер ПП «Наш продукт».
Члени комісії:	Марченко Светлана Миколаївна	головний технолог ПП «Наш продукт»;
	Христюк Володимир Володимирович	головній енергетик ПП «Наш продукт».

Склали дійсний акт впровадження про те, що результати дисертаційної роботи «Методи ідентифікації параметрів руху елементів каркасних конструкцій», представлені на здобуття вченого ступеня доктора філософії, впроваджені в виробничі процеси ПП «Наш продукт».

В результаті впровадження науково-дослідної роботи виконано:

а) впроваджено автоматизовану систему діагностики внутрішніх поверхонь промислових автоклавів із використанням СПМ як засобу позиціонування діагностичного інструменту, фотограмметричного методу сканування та алгоритму обробки хмари точок для виявлення потенційних дефектів поверхонь;

б) розроблено програмно-апаратну платформу керування СПМ на основі мікроконтролера Raspberry Pi Pico RP2040 з інерційним модулем MPU6050, яка реалізує методи комп'ютерного зору для дистанційного керування положенням діагностичного інструменту в режимі реального часу.

В результаті впровадження зазначених розробок отримано технічний ефект:

– підвищено якість та достовірність діагностики внутрішніх поверхонь автоклавів за рахунок застосування методу ідентифікації параметрів руху СПМ, що забезпечує точне позиціонування діагностичного інструменту в обмеженому просторі;

– скорочено час проведення діагностичних робіт завдяки автоматизації процесу сканування внутрішніх поверхонь автоклавів із використанням програмно-апаратної платформи керування СПМ;

– підвищено безпеку експлуатації автоклавного обладнання за рахунок своєчасного виявлення дефектів внутрішніх поверхонь на основі аналізу хмар точок, отриманих фотограмметричним методом.

Головний інженер ПП «Наш продукт»	Підпис	Окуневич І.М. ПІБ
Головний технолог ПП «Наш продукт»	Підпис	Марченко С.М. ПІБ
Головний енергетик ПП «Наш продукт»	Підпис	Христюк В.В. ПІБ

ДОДАТОК В

Лістинг комп’ютерна програма, «SPM API Server» (Сервер централізованого керування модулями обчислення параметрів руху сферичного паралельного механізму (SPM))

```
# ======== Лістинг програми «SPM API Server» ========
# Базова тека: SPM_API_server
# 🔒 Цей файл є захищеною версією лістингу для реєстрації авторських прав.
# 🚫 Запуск або використання коду у цьому вигляді неможливий.
# 🛡 Частину ключових алгоритмів приховано з метою захисту від несанкціонованого використання.
# Коментарі та пояснення вилучено для публічного представлення коду.

#
=========================================
=========
# === api/process_manager.py
#
=========================================
=========

import threading
import subprocess
import sys, os, runpy, json, shutil
from pathlib import Path
from common.project_manager import WorkDirs, ConfigManager
class ApiConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.3
    def __init__(
        self,
        config_name: str = "api_config.json",
        work_dirs=None,
        template_name: str = "api_config.json.default",
    ):
        self.work_dirs = work_dirs
        self.template_name = template_name
        base_dir = work_dirs.json_dir if work_dirs else "json"
        self.base_dir = base_dir
        self.config_path = Path(base_dir) / config_name
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.template_path = Path(base_dir) / template_name
        if not self.template_path.exists():
            self._save_default_config_template()
        self.ensure_config()

super().__init__(config_name=self.config_path.name, base_dir=self.base_dir)
        self.load()
    def _save_default_config_template(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_config = self._get_default_structure()
        with open(self.template_path, "w", encoding="utf-8") as f:
            json.dump(default_config, f, ensure_ascii=False, indent=4)
        print(f"[OK] Created default API template: {self.template_path}")
    def ensure_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        if self.config_path.exists():
            return
        if not self.template_path.exists():
            self._save_default_config_template()
        try:
            shutil.copyfile(self.template_path, self.config_path)
            with open(self.config_path, "r", encoding="utf-8") as f:
                self.data = json.load(f)
        except (json.JSONDecodeError, FileNotFoundError, OSError) as e:
            if self.work_dirs and self.work_dirs.is_debug:
                print(
                    f"[WARNING] template copy/load failed ({e}), writing default structure"
                )
            default_config = self._get_default_structure()
            os.makedirs(self.config_path.parent, exist_ok=True)
            with open(self.config_path, "w", encoding="utf-8") as f:
                json.dump(default_config, f, ensure_ascii=False, indent=4)
            self.data = default_config
            if self.work_dirs and self.work_dirs.is_debug:
                print(f"[DEBUG] Created config: {self.config_path}")
    def validate_structure(self):
        required_keys = {"server", "paths", "modules", "ui", "debug"}
        if not isinstance(self.data, dict):
            return False
        return required_keys.issubset(self.data.keys())
    def load(self):
        super().load()
        if not self.validate_structure():
            print(f"[WARNING] Invalid API config structure → reset to default")
            self.ensure_config()
            super().load()
```

```
        elif self.data.get("version", 0) < self.CURRENT_VERSION:
            print(f"[INFO] Updating API config version")
            self.data["version"] = self.CURRENT_VERSION
            self.save()
        return self.data
    def update_value(self, path_str: str, new_value, autosave: bool = True):
        """[пояснення вилучено для скороченого лістингу]"""
        keys = path_str.split(".")
        d = self.data
        for k in keys[:-1]:
            d = d[k]
        d[keys[-1]] = new_value
        if autosave:
            self.save()
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        return {
            "version": self.CURRENT_VERSION,
            "server": {
                "_comment": "⚙️ Параметри локального FastAPI сервера",
                "host": "0.0.0.0",
                "port": 8765,
                "log_level": "info",
                "auto_restart": True,
            },
            "paths": {
                "_comment": "📂 Шляхи до основних робочих і допоміжних тек",
                "default_work_dir": "",
                "modules_root": "./",
                "temp_dir": "./temp",
                "logs_dir": "./logs",
                "json_dir": "./json",
                "obj_dir": "./obj",
            },

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            "extensions": {
                "_comment": "🧱 Дозволені типи файлів для завантаження /upload",
                "allowed": [".obj", ".json"],
                "max_size_mb": 50,
            },
            "modules": {
                "_comment": "🧩 Опис активних модулів системи (можна вмикати/вимикати окремо)",
                "determ_work_area": {
                    "_comment": "Розрахунок робочої області",
                    "entry": "determ_work_area/determ_work_area.py",
                    "enabled": True,
                    "default_args": {
                        "input_obj": "",
                        "conv_out": "",
                        "valid_out": "",
                        "valid_res": "",
                        "output": "",
                        "unoutput": "",
                        "gen_new_path": False,
                        "is_new_path": False,
                        "is_validate": True,
                        "is_simpify": False,
                        "is_check": True,
                        "is_debug": False,
                    },
                },
                "getangles": {
                    "_comment": "Розрахунок кутів на основі точок або об’єктів",
                    "entry": "getangles/getangles.py",
                    "enabled": True,
                    "default_args": {
                        "is_bend": True,
                        "is_debug": False,
                        "is_multi": False,
                        "rc": 75.0,
                        "r1": 50.0,
                        "is_2d": False,
                        "is_3d": False,
                    },
                },
                "objtoangles": {
                    "_comment": "Перетворення OBJ моделей у кути або точки",
                    "entry": "objtoangles/objtoangles.py",
                    "enabled": True,
                    "default_args": {
                        "input": "",
                        "output": "",
                        "--unit": "rad",
                        "--unique-vertices": "unique",
                        "--sample": 0,
                        "--line-samples": 50,
                        "--filter": False,
                        "--clear": False,
                        "--is_debug": False,
                    },
                },
                "send_to_arduino": {
                    "_comment": "Передача даних у пристрій Arduino через COM-порт",
                    "entry": "send_to_arduino/to_arduino.py",
                    "enabled": False,
                },
                "ai_rtt": {
                    "_comment": "Модуль управління платформою через ComputerVision (AIVision) у режимі реального часу (RTT)",
                    "entry": "rtt_controllers/ai_rtt.py",
                    "enabled": False,
                },
                "joystick": {
                    "_comment": "Модуль управління платформою через Joystick у режимі реального часу (RTT)",
                    "entry": "rtt_controllers/joystick.py",
                    "enabled": False,
                },
            },
```

```
        "visualization": {
            "_comment": "🔍 Керування візуалізацією у віддаленому режимі",
            "disable_remote": True,
            "force_allow": False,
            "flags": ["is_simpify", "is_2d", "is_3d", "--filter"],
        },
        "ui": {
            "_comment": "🖥️ Параметри графічного інтерфейсу користувача (GUI)",
            "auto_start": True,
            "check_server_before_launch": True,
        },
        "debug": {
            "_comment": "🧠 Налагодження і логування",
            "is_debug": True,
            "log_to_file": True,
            "log_file": "server.log",
        },
        "ip": {"allowed": [], "disallowed": []},
    }
  def server(self):
    return self.data.get("server", {})
  def paths(self):
    return self.data.get("paths", {})
  def extensions(self):
    return self.data.get("extensions")
  def modules(self):
    return self.data.get("modules", {})

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

  def visualization(self):
    return self.data.get("visualization", {})
  def debug(self):
    return self.data.get("debug", {})
  def ui(self):
    return self.data.get("ui", {})
  def work_with_ip(self):
    return self.data.get("ip", {})
class ProcessManager:
  """[пояснення вилучено для скороченого лістингу]"""
  def __init__(self, work_dirs: WorkDirs | None = None):
    self.work_dirs = work_dirs
    self.config = ApiConfig(work_dirs=self.work_dirs)
    self.server_config = self.config.server()
    self.paths_config = self.config.paths()
    self.extension_config = self.config.extensions()
    self.modules_config = self.config.modules()
    self.visualization_config = self.config.visualization()
    self.debug_config = self.config.debug()
    self.ui_config = self.config.ui()
    self.work_with_ip = self.config.work_with_ip()
    self.active_processes = {}
    self.is_debug = self.debug_config["is_debug"]
  def _get_my_work_dir(self):
    return self.work_dirs.__repr__()
  def start_process(self, name: str, command: list[str]) -> None:
    """[пояснення вилучено для скороченого лістингу]"""
    if self.is_debug:
      print(f"[DEBUG] Starting process '{name}' → {' '.join(command)}")
    env = os.environ.copy()
    if self.work_dirs and hasattr(self.work_dirs, "base_dir"):
      env["PROJECT_ROOT"] = str(self.work_dirs.base_dir)
    env["SPM_SERVER_CHILD"] = "1"
    process = subprocess.Popen(
      command,
      stdout=subprocess.PIPE,
      stderr=subprocess.PIPE,
      text=True,
      env=env,
    )
    self.active_processes[name] = process
    if self.is_debug:
      def _stream_output(stream, prefix):
        for line in iter(stream.readline, ""):
          print(f"[{name}][{prefix}] {line.strip()}")
        stream.close()
      threading.Thread(
        target=_stream_output, args=(process.stdout, "OUT"), daemon=True
      ).start()
      threading.Thread(
        target=_stream_output, args=(process.stderr, "ERR"), daemon=True
      ).start()
    log_path = Path(self.work_dirs.base_dir) / f"logs/{name}.log"
    log_path.parent.mkdir(exist_ok=True)
    log_file = open(log_path, "w", encoding="utf-8")
    process = subprocess.Popen(
      command,
      stdout=subprocess.PIPE,
      stderr=subprocess.PIPE,
      text=True,
      env=env,
    )
    def _stream_output(stream, prefix):
      for line in iter(stream.readline, ""):
        print(f"[{name}][{prefix}] {line.strip()}")
        log_file.write(f"[{prefix}] {line}")
        log_file.flush()
      stream.close()
    threading.Thread(
      target=_stream_output, args=(process.stdout, "OUT"), daemon=True
    ).start()
    threading.Thread(
      target=_stream_output, args=(process.stderr, "ERR"), daemon=True
    ).start()
    if self.is_debug:
      print(f"[DEBUG] PID {process.pid} started for {name}")
  def check_status(self, name: str) -> dict[str, str | None] | str:
```

```
        """[пояснення вилучено для скороченого лістингу]"""
        process = self.active_processes.get(name)
        if not process:
            return {"status": "not_found", "exit_code": None}
        retcode = process.poll()
        if retcode is None:
            return {"status": "running", "exit_code": None}
        stderr = ""
        try:
            if process.stderr:
                stderr = process.stderr.read().strip()
        except Exception:
            pass
        status = "finished_ok" if retcode == 0 else "error"
        return {
            "status": status,
            "exit_code": retcode,
            "stderr": stderr,
        }
    def stop_process(self, name: str) -> None:
        """[пояснення вилучено для скороченого лістингу]"""
        process = self.active_processes.get(name)

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        if not process:
            print(f"[WARNING] Process '{name}' not found.")
            return
        process.terminate()
        process.wait(timeout=5)
        if self.is_debug:
            print(f"[DEBUG] Process '{name}' terminated.")
    def run_module(self, name: str, args: dict | None = None):
        """[пояснення вилучено для скороченого лістингу]"""
        module = self.modules_config.get(name)
        if not module:
            print(f"[ERROR] Module '{name}' not found in api_config.json.")
            return
        if not module.get("enabled", True):
            print(f"[INFO] Module '{name}' is disabled in api_config.json.")
            return
        entry = module.get("entry")
        if not entry:
            print(f"[ERROR] Module '{name}' has no entry path defined.")
            return
        candidate = Path(self.work_dirs.base_dir) / entry
        internal_candidate = Path(self.work_dirs.base_dir) / "_internal" / entry
        script_path = candidate if candidate.exists() else internal_candidate
        if not script_path.exists():
            print(f"[ERROR] Script {script_path} not found.")
            return
        vis_cfg = self.visualization_config or {}
        disable_remote = vis_cfg.get("disable_remote", True)
        force_allow = vis_cfg.get("force_allow", False)
        vis_flags = vis_cfg.get("flags", [])
        use_defaults = bool(args and args.get("default_args", False))
        if use_defaults:
            args = module.get("default_args", {}).copy()
            print(f"[INFO] Використовуються default_args для '{name}' → {args}")
        elif args:
            args.pop("default_args", None)
        else:
            args = {}
        is_remote = (
            not sys.stdin.isatty()
            or os.environ.get("SSH_CONNECTION")
            or os.environ.get("REMOTE_CONSOLE")
        )
        if disable_remote and not force_allow and is_remote:
            print("[WARN] Візуалізація недоступна для віддаленого запуску.")
            for k in list(args.keys()):
                if any(
                    flag.lower().lstrip("-") == k.lower().lstrip("-")
                    for flag in vis_flags
                ):
                    args[k] = False
            args["_warning"] = "Візуалізація недоступна для віддаленого запуску."
        cli_args = []
        for k, v in args.items():
            if k.startswith("_"):
                continue
            if k in ("input", "output", "alpha", "beta", "gamma"):
                cli_args.append(str(v))
            elif isinstance(v, bool):
                cli_args.append(f"--{k}" if v else f"--no-{k}")
            else:
                cli_args.extend([f"--{k}", str(v)])
        if getattr(sys, "frozen", False):
            print(f"[INFO] ▶ Виконуємо '{name}' inline (всередині поточного процесу).")
            argv_backup = sys.argv.copy()
            sys.argv = [str(script_path)] + cli_args
            script_dir = os.path.dirname(str(script_path))
            if script_dir not in sys.path:
                sys.path.insert(0, script_dir)
            try:
                runpy.run_path(str(script_path), run_name="__main__")
            finally:
                sys.argv = argv_backup
            return
        cmd = [sys.executable, str(script_path)] + cli_args
        self.start_process(name, cmd)
    def run_all_enabled_modules(self):
        for name, module in self.modules_config.items():
```

```
            if module.get("enabled", False):
                self.run_module(name)
    def cleanup(self):
        """[пояснення вилучено для скороченого лістингу]"""
        for name, process in self.active_processes.items():
            if process.poll() is None:
                process.terminate()
                if self.is_debug:
                    print(f"[DEBUG] Terminated process '{name}' (PID {process.pid}).")
    def shutdown(self):
        """[пояснення вилучено для скороченого лістингу]"""
        for name, process in self.active_processes.items():
            if process.poll() is None:
                print(f"Stopping {name}...")
                process.terminate()
                try:
                    process.wait(timeout=5)
                except subprocess.TimeoutExpired:
                    print(f"[FORCE] Killing {name}...")
                    process.kill()
    def get_status_report(self) -> dict:

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        report = {}
        for name, process in self.active_processes.items():
            status = "running" if process.poll() is None else "finished"
            report[name] = {"pid": process.pid, "status": status}
        return report
    def check_ip_access(self, request) -> bool:
        """[пояснення вилучено для скороченого лістингу]"""
        client_ip = None
        try:
            client_ip = request.client.host if request.client else None
        except Exception:
            pass
        allowed = self.work_with_ip.get("allowed", [])
        disallowed = self.work_with_ip.get("disallowed", [])
        if client_ip in disallowed:
            print(f"[ACCESS DENIED] {client_ip} заблокований у disallowed.")
            return False
        if allowed:
            if client_ip in allowed or client_ip in ("127.0.0.1", "::1", "localhost"):
                return True
            print(f"[ACCESS DENIED] {client_ip} не в списку allowed.")
            return False
        if client_ip in ("127.0.0.1", "::1", "localhost"):
            return True
        print(f"[ACCESS DENIED] Віддалений запит з {client_ip} відхилено.")
        return False
#
================================================
# === api/server.py
#
================================================

import os, sys, threading, time
from datetime import datetime
import socket
from fastapi import FastAPI, HTTPException, Request, UploadFile, File
from fastapi.responses import JSONResponse
import uvicorn
from pathlib import Path
from api.process_manager import ProcessManager
from common.project_manager import WorkDirs
def _set_my_work_dir():
    """[пояснення вилучено для скороченого лістингу]"""
    #♡ Код заблоковано у захищеній версії лістингу.

base_dir = _set_my_work_dir()
work_dirs = WorkDirs(base_dir=base_dir)
manager = ProcessManager(work_dirs=work_dirs)
app = FastAPI(
    title="SPM API Server",
    description=("API для керування модулями SPM-системи."),
    version="0.2",
    docs_url="/__dev_docs",
    redoc_url="/__redoc_open_docs",
    openapi_url="/__openapi_docs.json",
)
@app.middleware(
    "http"
)
async def protect_docs(request: Request, call_next):
    if (
        request.url.path.startswith("/__dev_docs")
        or request.url.path.startswith("/__redoc_open_docs")
        or request.url.path.startswith("/__openapi_docs.json")
    ):
        token = request.headers.get("X-Internal-Token")
        if token != "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx":
            raise HTTPException(status_code=403, detail="Access denied")
    return await call_next(request)

@app.post("/__get_my_work_dir", include_in_schema=False)
def _get_my_work_dir(request: Request):
    if not manager.check_ip_access(request):
        raise HTTPException(status_code=403, detail="Access denied (IP restricted)")
    prog_dir = manager._get_my_work_dir()
    return {prog_dir}
```

```
@app.post("/api/__testing_is_running",
include_in_schema=False)
def test():
    """[пояснення вилучено для скороченого лістингу]"""
    return {"status": "ok", "message": "SPM API is running"}
@app.post("/api/__shutdown",
include_in_schema=False)
def shutdown(request: Request):
    """[пояснення вилучено для скороченого лістингу]"""
    if not manager.check_ip_access(request):
        raise HTTPException(status_code=403, detail="Access denied (IP restricted)")
    print("[INFO] Отримано запит на завершення роботи сервера.")
    manager.shutdown()
    def delayed_exit():
        time.sleep(0.5)
        print("✅ Сервер успішно зупинено.")
        os._exit(0)
    threading.Thread(target=delayed_exit, daemon=True).start()
    return {"status": "server_shutdown"}
@app.get("/api/ping")
async def ping():
    """[пояснення вилучено для скороченого лістингу]"""
    return {
        "ok": True,
        "server": "SPM API Server",
        "version": "1.0.0",
        "status": "running",
        "ip": socket.gethostbyname(socket.gethostname()),
        "port": 8765,
        "time": datetime.utcnow().isoformat() + "Z",
    }
@app.post("/api/upload")
async def upload_file(file: UploadFile = File(...)):
    cfg_ext = manager.extension_config.get("extensions", {})
    allowed_exts = set(cfg_ext.get("allowed", [".obj", ".json"]))
    max_size_mb = cfg_ext.get("max_size_mb", 10)
    ext = Path(file.filename).suffix.lower()
    if ext not in allowed_exts:
        raise HTTPException(
            status_code=400,

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            detail=f"Недопустимий тип файлу '{ext}'. Дозволено тільки: {', '.join(allowed_exts)}",
        )
    safe_name = os.path.basename(file.filename)
    dest = Path(work_dirs.upload_dir) / safe_name
    contents = await file.read()
    size_mb = len(contents) / (1024 * 1024)
    if size_mb > max_size_mb:
        raise HTTPException(status_code=400, detail=f"Файл перевищує {max_size_mb} МБ")
    with open(dest, "wb") as f:
        f.write(contents)
    relative_path = os.path.relpath(dest, start=work_dirs.base_dir)
    return {"uploaded_to": relative_path, "size_mb": round(size_mb, 2)}
@app.get("/api/modules")
def list_modules():
    """[пояснення вилучено для скороченого лістингу]"""
    return manager.modules_config
@app.get("/api/status")
def get_status():
    """[пояснення вилучено для скороченого лістингу]"""
    report = manager.get_status_report()
    return JSONResponse(content=report)
@app.post("/api/start/{module_name}")
async def start_module(module_name: str, request: Request):
    """[пояснення вилучено для скороченого лістингу]"""
    modules = manager.modules_config
    if module_name not in modules:
        raise HTTPException(
            status_code=404, detail=f"Модуль '{module_name}' не знайдено"
        )
    module = modules[module_name]
    if not module.get("enabled", True):
        raise HTTPException(status_code=403, detail=f"Модуль '{module_name}' вимкнено")
    try:
        payload = await request.json()
    except Exception:
        payload = {}
    manager.run_module(module_name, args=payload)
    return {
        "status": "started",
        "module": module_name,
        "args": payload,
        "entry": module.get("entry"),
    }
@app.post("/api/stop/{module_name}")
def stop_module(module_name: str):
    """[пояснення вилучено для скороченого лістингу]"""
    manager.stop_process(module_name)
    return {"status": "stopped", "module": module_name}
@app.post("/api/restart/{module_name}")
def restart_module(module_name: str):
    """[пояснення вилучено для скороченого лістингу]"""
    manager.stop_process(module_name)
    modules = manager.config.modules()
    if module_name in modules:
        entry = modules[module_name].get("entry")
        manager.start_process(module_name, ["python3", entry])
        return {"status": "restarted", "module": module_name}
```

```
    raise HTTPException(status_code=404, detail="Модуль не знайдено")
@app.post("/api/start_all", include_in_schema=False)
def start_all():
    """[пояснення вилучено для скороченого лістингу]"""
    manager.run_all_enabled_modules()
    return {"status": "started_all"}
@app.post("/api/stop_all")
def stop_all():
    """[пояснення вилучено для скороченого лістингу]"""
    manager.shutdown()
    return {"status": "all stopped"}
@app.get("/api/config")
def get_config():
    """[пояснення вилучено для скороченого лістингу]"""
    return JSONResponse(content=manager.config.data)
if __name__ == "__main__":
    server_config = manager.config.server.get()
    host = server_config.get("host")
    port = server_config.get("port")
    uvicorn.run("api.server:app", host=host, port=port, reload=False)

#
========================================
=========
# === common/project_manager.py
#
========================================
=========

import sys, os, json
import tempfile
from pathlib import Path
def get_default_work_dirs(is_debug: bool = False):
    """[пояснення вилучено для скороченого лістингу]"""
    env_root = os.environ.get("PROJECT_ROOT")
    work_dirs = WorkDirs(base_dir=env_root, is_debug=is_debug)
    return work_dirs

class WorkDirs:
    def __init__(self, base_dir: str, is_debug: bool = False):
        self.is_debug = is_debug
        self.base_dir = os.path.abspath(base_dir)
        if "MEI" in self.base_dir:
            print(f"[WARN] WorkDirs base_dir looks temporary: {self.base_dir}")
        self.json_dir = os.path.join(self.base_dir, "json")
        self.obj_dir = os.path.join(self.base_dir, "obj")
        self.upload_dir = os.path.join(self.base_dir, "upload")
        os.makedirs(self.json_dir, exist_ok=True)
        os.makedirs(self.obj_dir, exist_ok=True)
        os.makedirs(self.upload_dir, exist_ok=True)
        if self.is_debug:
            print(f"[DEBUG] WorkDirs created in {self.base_dir}")
    def __repr__(self):
        return f"WorkDirs(base={self.base_dir})"
class ConfigManager:
    """[пояснення вилучено для скороченого лістингу]"""
    def __init__(self, config_name: str, base_dir: str = "json"):
        self.data = {}
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.config_path = self.config_base_dir / config_name
        if self.config_path.exists():
            self.load()
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(self.config_path, "r", encoding="utf-8") as f:
            self.data = json.load(f)
    def save(self):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(self.config_path, "w", encoding="utf-8") as f:
            json.dump(self.data, f, ensure_ascii=False, indent=4)
    def get(self, *keys, default=None):
        """[пояснення вилучено для скороченого лістингу]"""
        node = self.data
        for key in keys:
            if isinstance(node, dict) and key in node:
                node = node[key]
            else:
                return default
        return node
    def set(self, *keys, value):
        """[пояснення вилучено для скороченого лістингу]"""
        node = self.data
        for key in keys[:-1]:
            node = node.setdefault(key, {})
        node[keys[-1]] = value
    def __repr__(self):
        return f"ConfigManager({self.config_path.name}, keys={len(self.data)})"

#
========================================
=========
# === determ_work_area/determ_work_area.py
#
========================================
=========

import argparse, sys, json
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
from pathlib import Path
from concurrent.futures import ProcessPoolExecutor, as_completed
from getangles.getangles import GetAngles
from objtoangles.objtoangles import ObjToAngles
```

```
from common.project_manager import ConfigManager,
get_default_work_dirs
class DetermStatic:
    @staticmethod
    def _generate_new_path(output_file=None):
        """[пояснення вилучено для скороченого
лістингу]"""
        if not output_file:
            output_file = "../obj/points.obj"
        x_min, x_max = -100, 100
        y_min, y_max = -100, 100
        z_min, z_max = 0, 100
        step = 5
        count = 0
        with open(output_file, "w", encoding="utf-8") as f:
            for z in range(z_min, z_max + 1, step):
                for y in range(y_min, y_max + 1, step):
                    for x in range(x_min, x_max + 1, step):
                        f.write(f"v {x} {z} {y}\n")
                        print(f"v {x} {z} {y}\n")
                        count += 1
        print(f"[OK] Файл '{output_file}' створено
({count:,} точок)")
    @staticmethod
    def process_angle(point, angle, is_debug=False):
        """[пояснення вилучено для скороченого
лістингу]"""
        alpha, beta, gamma = angle["alpha"],
angle["beta"], angle["gamma"]
        x, y, z = point["x"], point["y"], point["z"]
        servo_angles = GetAngles(
            is_debug=is_debug,
            is_multi=True,
        ).calculation(alpha=alpha, beta=beta,
gamma=gamma)
        valid = not all(np.isnan(value) for value in
servo_angles)
        result_entry = {
            "point": {"x": x, "y": y, "z": z},
            "angle": {"alpha": alpha, "beta": beta, "gamma":
gamma},
            "servo": {
                "s1": servo_angles[0],
                "s2": servo_angles[1],
                "s3": servo_angles[2],
            },
            "valid": valid,
        }
        return result_entry
    @staticmethod
    def filter_and_visualize_points(
        input_json="filtered.json",
output_json="valid_result.json"
    ):
        """[пояснення вилучено для скороченого
лістингу]"""
        with open(input_json, "r", encoding="utf-8") as f:
            data = json.load(f)
        points = np.array(
            [[p["x"], p["y"], p["z"]] for p in data["points"]],
dtype=float
        )
        angles = np.array(
            [[a["alpha"], a["beta"], a["gamma"]] for a in
data["angles"]], dtype=float
        )
        if len(points) < 4:
            print("Недостатньо точок для побудови
опуклої оболонки.")
            return
        hull = ConvexHull(points)
        surface_indices = np.unique(hull.vertices)
        valid_points = points[surface_indices]
        valid_angles = angles[surface_indices]
        result = {
            "points": [
                {"x": float(x), "y": float(y), "z": float(z)}
                for x, y, z in valid_points
            ],
            "angles": [
                {"alpha": float(alpha), "beta": float(beta),
"gamma": float(gamma)}
                for alpha, beta, gamma in valid_angles
            ],
        }
        with open(output_json, "w", encoding="utf-8") as
f_out:
            json.dump(result, f_out, ensure_ascii=False,
indent=2)
        print(f"[OK] Збережено {len(valid_points)}
точок у {output_json}")
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        xs, ys, zs = valid_points[:, 0], valid_points[:, 1],
valid_points[:, 2]
        ax.scatter(xs, ys, zs, color="red", s=50,
label="Valid points")
        # З'єднуємо всі точки між собою лініями
        for i in range(len(valid_points)):
            for j in range(i + 1, len(valid_points)):
                ax.plot(
                    [valid_points[i, 0], valid_points[j, 0]],
                    [valid_points[i, 1], valid_points[j, 1]],
                    [valid_points[i, 2], valid_points[j, 2]],
                    color="blue",
                    linewidth=0.5,
                    alpha=0.4,
                )
        ax.set_xlabel("X")
        ax.set_ylabel("Y")
        ax.set_zlabel("Z")

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

        ax.legend()
        plt.title("Робоча область — зовнішні точки,
з'єднані між собою")
        plt.show()
class AreaConfig(ConfigManager):
    """[пояснення вилучено для скороченого
лістингу]"""
    CURRENT_VERSION = 0.1
    def __init__(self, config_name="area_config.json",
work_dirs=None, is_debug=False):
        self.work_dirs = work_dirs
        self.is_debug = is_debug
```

```
        base_dir = work_dirs.json_dir if work_dirs else
"json"
        super().__init__(config_name, base_dir)
        if not self.config_path.exists():
            self.ensure_default_config()
    def ensure_default_config(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        default_data = {
            "version": self.CURRENT_VERSION,
            "last_used": {
                "input_obj": "../obj/points.obj",
                "conv_out": "../json/conv_output.json",
                "valid_out": "../json/valid.json",
                "valid_res": "../json/valid_result.json",
                "output": "../json/output.json",
                "unoutput": "../json/unoutput.json",
            },
            "stats": {},
        }
        self.config_base_dir.mkdir(parents=True,
exist_ok=True)
        with open(self.config_path, "w", encoding="utf-
8") as f:
            json.dump(default_data, f, ensure_ascii=False,
indent=4)
        self.data = default_data
        if self.is_debug:
            print(f"[DEBUG] Created default area config:
{self.config_path}")
    @staticmethod
    def load_data(path):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        if not path.exists():
            raise FileNotFoundError(f"JSON file not found:
{path}")
        with open(path, "r", encoding="utf-8") as f:
            return json.load(f)
    @staticmethod
    def save_json(path, data):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False,
indent=2)
        print(f"[OK] Saved JSON → {path}")
    @staticmethod
    def save_obj(path, vertices):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, "w", encoding="utf-8") as f:
            f.write("\n".join(vertices))
        print(f"[OK] Saved OBJ ({len(vertices)} points) →
{path}")
class DetermWorkArea:
    def __init__(
        self,
        input_obj: str = "points.obj",
        conv_out: str = "conv_output.json",
        valid_out: str = "valid.json",
        valid_res: str = "valid_result.json",
        output: str = "output.json",
        unoutput: str = "unoutput.json",
        *,
        gen_new_path: bool = False,
        is_new_path: bool = False,
        is_validate: bool = True,
        is_simpify: bool = False,
        is_check: bool = True,
        is_debug: bool = False,
    ):
        super().__init__()
        self.gen_new_path = gen_new_path
        self.is_new_path = is_new_path
        self.is_validate = is_validate
        self.is_simpify = is_simpify
        self.is_check = is_check
        self.is_debug = is_debug
        self.work_dirs = get_default_work_dirs()
        self.json_dir = Path(self.work_dirs.json_dir)
        self.obj_dir = Path(self.work_dirs.obj_dir)
        self.input_obj_path = str(self.obj_dir / input_obj)
        self.conv_out_path = str(self.json_dir / conv_out)
        self.valid_out_path = str(self.json_dir / valid_out)
        self.valid_res = str(self.json_dir / valid_res)
        self.output_path = str(self.json_dir / output)
        self.unoutput_path = str(self.json_dir / unoutput)
        self.ac = AreaConfig(work_dirs=self.work_dirs,
is_debug=self.is_debug)
        if self.gen_new_path:

DetermStatic._generate_new_path(self.input_obj_path)
        if self.is_new_path:
            self._convert_obj_to_angles()
        if self.is_validate:
            self._validate_angles()
        if self.is_simpify:
            DetermStatic.filter_and_visualize_points(

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

                self.valid_out_path, self.valid_res
            )
        if self.is_check:
            self.check_perm_angles()
    def _convert_obj_to_angles(self):
        converter = ObjToAngles(
            self.input_obj_path,          self.conv_out_path,
is_filter=False
        )
        converter.run()
    def _validate_angles(self):
        data = self.ac.load_data(self.conv_out_path)
        points, angles = data["points"], data["angles"]
        result = {"points": [], "angles": []}
        for point, angle in zip(points, angles):
            if (
                any(abs(angle[k]) > 46 for k in ["alpha",
"beta"])
                or abs(angle["gamma"]) > 180
            ):
```

```
            continue
        result["points"].append(point)
        result["angles"].append(angle)
        if self.is_debug:
            print(f"{point} → {angle}")
    self.ac.save_json(self.valid_out_path, result)
  def check_perm_angles(self):
    result = {"points": [], "angles": [], "servo": []}
    unresult = {"points": [], "angles": [], "servo": []}
    obj_out_file = []
    obj_unout_file = []
    path = self.valid_res if self.is_simpify else self.valid_out_path
    data = self.ac.load_data(path)
    points, angles = data["points"], data["angles"]
    with ProcessPoolExecutor() as executor:
        futures = [
            executor.submit(DetermStatic.process_angle, point, angle, self.is_debug)
            for point, angle in zip(points, angles)
        ]
        for future in as_completed(futures):
            entry = future.result()
            x, y, z = entry["point"].values()
            if entry["valid"]:
                result["points"].append(entry["point"])
                result["angles"].append(entry["angle"])
                result["servo"].append(entry["servo"])
                obj_out_file.append(f"v {x:.3f} {z:.3f} {y:.3f}")
            else:
                unresult["points"].append(entry["point"])
                unresult["angles"].append(entry["angle"])
                unresult["servo"].append(entry["servo"])
                obj_unout_file.append(f"v {x:.3f} {z:.3f} {y:.3f}")
    self.ac.save_json(self.output_path, result)
    self.ac.save_json(self.unoutput_path, unresult)
    self.ac.save_obj(self.obj_dir / "obj_work_area.obj", obj_out_file)
    self.ac.save_obj(self.obj_dir / "obj_unwork_area.obj", obj_unout_file)
  @staticmethod
  def build_parser():
    epilog_text = """[пояснення вилучено для скороченого лістингу]"""
    parser = argparse.ArgumentParser(
        prog="determ_work_area",
        description=(
            "Комплексний розрахунок робочої зони SPM: "
            "генерація, конвертація OBJ→JSON, валідація та обчислення кутів сервоприводів."
        ),
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog=epilog_text,
    )
    parser.add_argument(
        "--input_obj",
        type=str,
        default="points.obj",
        help="Вхідний OBJ файл (у директорії obj/).",
    )
    parser.add_argument(
        "--conv_out",
        type=str,
        default="conv_output.json",
        help="Вихідний JSON після конвертації OBJ→ANGLES.",
    )
    parser.add_argument(
        "--valid_out",
        type=str,
        default="valid.json",
        help="Файл відфільтрованих точок після перевірки кутів α, β, γ.",
    )
    parser.add_argument(
        "--valid_res",
        type=str,
        default="valid_result.json",
        help="Файл результатів після спрощення (simplify).",
    )
    parser.add_argument(
        "--output",
        type=str,
        default="output.json",
        help="Файл робочих точок (успішні розрахунки).",
    )
    parser.add_argument(
        "--unoutput",
        type=str,
        default="unoutput.json",
        help="Файл неробочих точок (невдалі розрахунки).",
    )
    parser.add_argument(
        "--gen_new_path",
        action=argparse.BooleanOptionalAction,
        default=False,
        help="Генерувати новий OBJ шлях для тесту.",
    )
    parser.add_argument(
        "--is_new_path",
        action=argparse.BooleanOptionalAction,
        default=False,
        help="Конвертувати OBJ→ANGLES через ObjToAngles.",
    )
    parser.add_argument(
        "--is_validate",
        action=argparse.BooleanOptionalAction,
        default=True,
        help="Фільтрувати кути αβγ після конвертації.",
    )
    parser.add_argument(
        "--is_simpify",
        action=argparse.BooleanOptionalAction,
```

```
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---
```

```
            default=False,
            help="Спрощувати результати після валідації (викликає filter_and_visualize_points).",
        )
        parser.add_argument(
            "--is_check",
            action=argparse.BooleanOptionalAction,
            default=True,
            help="Виконати розрахунок кутів сервоприводів (через GetAngles).",
        )
        parser.add_argument(
            "--is_debug",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="[DEBUG_INFO] Виводити проміжні повідомлення.",
        )
        return parser
    @staticmethod
    def run_terminal():
        parser = DetermWorkArea.build_parser()
        if len(sys.argv) == 1:
            parser.print_help()
            sys.exit(0)
        args = parser.parse_args()
        app = DetermWorkArea(
            input_obj=args.input_obj,
            conv_out=args.conv_out,
            valid_out=args.valid_out,
            valid_res=args.valid_res,
            output=args.output,
            unoutput=args.unoutput,
            gen_new_path=args.gen_new_path,
            is_new_path=args.is_new_path,
            is_validate=args.is_validate,
            is_simpify=args.is_simpify,
            is_check=args.is_check,
            is_debug=args.is_debug,
        )
        if args.is_debug:
            print("[OK] DetermWorkArea finished successfully.")
        return app
if __name__ == "__main__":
    if len(sys.argv) > 1:
        DetermWorkArea.run_terminal()
    else:
        app = DetermWorkArea(
            gen_new_path=True,
            is_new_path=True,
            is_validate=True,
            is_simpify=False,
            is_check=True,
            is_debug=False,
        )

#
================================================
=========
# === getangles/__init__.py
#
================================================
=========

"""[пояснення вилучено для скороченого лістингу]"""

#
================================================
=========
# === getangles/getangles.py
#
================================================
=========

import os, sys, re, json, shutil
import argparse
import numpy as np
from numpy import arcsin
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
import itertools
from mpl_toolkits.mplot3d import Axes3D
from pathlib import Path
from common.project_manager import ConfigManager, get_default_work_dirs
"""[пояснення вилучено для скороченого лістингу]"""
class MathFunction:
    @staticmethod
    def sin_alpha(a, b):
        """[пояснення вилучено для скороченого лістингу]"""
        return (a / b**2) * np.sqrt(b**2 - (a**2) / 4)
    @staticmethod
    def get_vector(bx0, by0, bx, by):
        """[пояснення вилучено для скороченого лістингу]"""
        if bx0 == bx and by0 == bx:
            return 0
        return np.sqrt((bx - bx0) ** 2 + (by - by0) ** 2)
class AnglesConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.1
    def __init__(
        self,
        config_name="angles_config.json",
        work_dirs=None,
        template_name="angles_config.json.default",
        is_multi: bool = False,
    ):
        self.work_dirs = work_dirs
        self.template_name = template_name
        self.prefix = "angles_config"
        self.max_attempts = 10000
        self.is_multi = is_multi
        base_dir = work_dirs.json_dir if work_dirs is not None else "json"
        self.base_dir = base_dir
        self.config_path = Path(base_dir) / config_name
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.template_path = self.config_base_dir / self.template_name
        if not self.template_path.exists():
            self._save_default_config_template()
        if self.is_multi:
```

```
            self.create_instance_config()
        else:
            self.ensure_config()
        super().__init__(
            config_name=self.config_path.name,
base_dir=str(self.config_base_dir)
        )
        self.load()
    def _save_default_config_template(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_config = self._get_default_structure()
        with open(self.template_path, "w", encoding="utf-8") as f:
            json.dump(default_config, f, ensure_ascii=False, indent=4)
        if self.work_dirs and self.work_dirs.is_debug:
            print(f"[DEBUG] Created default config: {self.config_path}")
    def ensure_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        if self.config_path.exists():
            return
        if not self.template_path.exists():
            self._save_default_config_template()
        try:
            shutil.copyfile(self.template_path, self.config_path)
            with open(self.config_path, "r", encoding="utf-8") as f:
                self.data = json.load(f)
        except (json.JSONDecodeError, FileNotFoundError, OSError) as e:
            if self.work_dirs and self.work_dirs.is_debug:
                print(
                    f"[WARNING] template copy/load failed ({e}), writing default structure"
                )
            default_config = self._get_default_structure()
            with open(self.config_path, "w", encoding="utf-8") as f:
                json.dump(default_config, f, ensure_ascii=False, indent=4)
            self.data = default_config
            if self.work_dirs and self.work_dirs.is_debug:
                print(f"[DEBUG] Created config: {self.config_path}")
    def validate_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        required_keys = {
            "version",
            "coordinates",
            "angles",
        }
        if not isinstance(self.data, dict):
            return False
        if not required_keys.issubset(self.data.keys()):
            return False
        version = self.data.get("version")
        if not isinstance(version, (float, int)):
            return False
        try:
            _ = self.data["angles"]["platform"]["zero"]
            _ = self.data["coordinates"]["platform"]["zero"]
        except (KeyError, TypeError):
            return False

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        return True
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        super().load()
        if not self.validate_structure():
            print(
                f"[WARNING] Invalid config structure in {self.config_path}, resetting..."
            )
            self.ensure_config()
        elif self.data["version"] < self.CURRENT_VERSION:
            print(
                f"[INFO] Updating config version from {self.data['version']} to {self.CURRENT_VERSION}"
            )
            self.data["version"] = self.CURRENT_VERSION
            self.save()
    def create_instance_config(self):
        if not self.template_path.exists():
            self._save_default_config_template()
        with open(self.template_path, "rb") as f:
            template_bytes = f.read()
        pattern = re.compile(rf"^{re.escape(self.prefix)}\.(\d+)\.json$")
        existing = [
            int(m.group(1))
            for fname in os.listdir(self.config_base_dir)
            if (m := pattern.match(fname))
        ]
        next_number = max(existing) + 1 if existing else 1
        for _ in range(self.max_attempts):
            dest_name = f"{self.prefix}.{next_number}.json"
            dest_path = self.config_base_dir / dest_name
            try:
                with open(dest_path, "xb") as out:
                    out.write(template_bytes)
                self.config_path = dest_path
                with open(dest_path, "r", encoding="utf-8") as f:
                    self.data = json.load(f)
                if self.work_dirs and self.work_dirs.is_debug:
                    print(f"[DEBUG] Created new config: {dest_path}")
                return dest_path
            except FileExistsError:
                next_number += 1
        raise RuntimeError(
            f"Unable to create unique config file after {self.max_attempts} attempts"
        )
    def update_value(self, path_str, new_value):
        keys = path_str.split(".")
```

```
        d = self.data
        for k in keys[:-1]:
            d = d[k]
        d[keys[-1]] = new_value
        self.save()
        if self.work_dirs and self.work_dirs.is_debug:
            print(f"[DEBUG] Updated {path_str} =
{new_value}")
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        return {
            "version": self.CURRENT_VERSION,
            "coordinates": {
                "platform": {
                    "zero": {
                        "left": {
                            "c": [40.0, 0.0, 55.0],
                            "c2": [-20, -34.64, 55.0],
                            "c3": [-20, 34.64, 55.0],
                        },
                        "right": {
                            "c": [40.0, 0.0, 55.0],
                            "c2": [-20, -34.64, 55.0],
                            "c3": [-20, 34.64, 55.0],
                        },
                    },
                    "last": {},
                    "all": [],
                },
                "middle": {
                    "zero": {
                        "left": {
                            "b": [18.75, -46.35124054],
                            "b2": [-49.5161937, 6.93877234],
                            "b3": [30.7662487, 39.41367708],
                        },
                        "right": {
                            "b": [18.75, 46.35124054],
                            "b2": [30.7662487, -39.41367708],
                            "b3": [-49.5161937, -6.93877234],
                        },
                    },
                    "last": {},
                    "all": [],
                },
            },
            "angles": {
                "platform": {
                    "zero": {"alpha": 0, "beta": 0, "gamma": 0},
                    "last": {},
                    "all": [],
                },
                "servo": {
                    "zero": {"s1": 90, "s2": 90, "s3": 90},

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

                    "last": {},
                    "all": [],
                },
            },
        }
class GetAngles:
    def __init__(
        self,
        *,
        is_bend: bool = True,
        is_debug: bool = False,
        is_multi: bool = False,
        rc: float = 75.0,
        r1: float = 50.0,
        is_2d: bool = False,
        is_3d: bool = False,
    ):
        self.work_dirs = get_default_work_dirs()
        self.Rc = rc
        self.R1 = r1
        self.is_bend = is_bend
        self.is_debug = is_debug
        self.is_multi = is_multi
        self.is_2d = is_2d
        self.is_3d = is_3d
        self.alpha = self.beta = self.gamma = 0.0
        self.is_p_last_coords = False
        self.is_m_last_coords = False
        self.is_p_last_angles = False
        self.is_s_last_angles = False
        self.flag_map = {
            "coords": {
                "platform": "is_p_last_coords",
                "middle": "is_m_last_coords",
            },
            "angles": {
                "platform": "is_p_last_angles",
                "servo": "is_s_last_angles",
            },
        }
        self.get_config()
        self.p_coords, self.m_coords, self.candidates = [],
[], []
    def calculation(self, alpha=None, beta=None,
gamma=None):
        if alpha is not None:
            self.alpha = alpha
        if beta is not None:
            self.beta = beta
        if gamma is not None:
            self.gamma = gamma
        if not self.is_multi:
            self.p_coords.clear()
            self.m_coords.clear()
            self.candidates.clear()
        self.p_last_coords = self.get_p_last_coords()
        self.m_last_coords = self.get_m_last_coords()
        self.p_last_angles = self.get_p_last_angles()
        self.s_last_angles = self.get_s_last_angles()
        for i, p_last_coord in
enumerate(self.p_last_coords):
            p_coord =
self.get_p_coords(coords=p_last_coord)
            m_coord = self.get_m_cords(p_coord)
            if m_coord is None:
                if self.is_debug:
                    print(
```

```
                f"⚠️ Пропущено розрахунок для поточного
                f"(α,β,γ) = ({self.alpha}, {self.beta}, {self.gamma}) — "
                "отримано уявні або вироджені координати при побудові middle."
            )
            return (np.nan, np.nan, np.nan)
        bx1, bx2, by1, by2 = m_coord
        self.p_coords.append(p_coord)
        self.m_coords.append(m_coord)
        self.candidates.append([(bx1, by1), (bx1, by2), (bx2, by1), (bx2, by2)])
    if self.is_debug:
        print(self.candidates[0])
        print(self.candidates[1])
        print(self.candidates[2])
    valid_combos = [
        self.filter_points_on_circle(cand, self.R1) for cand in self.candidates
    ]
    if not all(valid_combos):
        if self.is_debug:
            sizes = [len(v) for v in valid_combos]
            print(
                "⚠️ Після перевірки належності до кола одна з груп кандидатів пуста — "
                f"valid sizes = {sizes}. Пропускаємо набір alpha/beta/gamma."
            )
        return (np.nan, np.nan, np.nan)
    valid_combos = self.filter_valid_combinations(tuple(valid_combos), min_dist=10)
    if not valid_combos:
        if self.is_debug:
            print("⚠️ Після фільтрації за відстанню — valid_combos порожній.")
        return (np.nan, np.nan, np.nan)
    if self.is_2d:
        self.plot_combinations(
            valid_combos,
            self.p_coords,
            decimals=2,
        )
    valid_combos = [

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        combo
        for combo in valid_combos
        if self.is_valid_combo(
            combo[0],
            combo[1],
            combo[2],
            self.p_coords,
        )
    ]
    if not valid_combos:
        if self.is_debug:
            print("⚠️ Після перевірки перетинів — valid_combos порожній.")
            return (np.nan, np.nan, np.nan)
    if self.is_2d:
        self.plot_combinations(
            valid_combos,
            self.p_coords,
            decimals=2,
        )
    valid_combos = self.filter_by_bend_side(valid_combos, self.p_coords)
    closest_combo = self.choose_closest_combo(
        valid_combos,
        self.m_last_coords,
    )
    valid_combos = [closest_combo] if closest_combo else []
    if not valid_combos:
        if self.is_debug:
            print("⚠️ Після перевірки перетинів — valid_combos порожній.")
        return (np.nan, np.nan, np.nan)
    if self.is_2d:
        self.plot_combinations(
            valid_combos,
            self.p_coords,
            decimals=2,
        )
    a, a2, a3 = self.get_angles(
        self.m_last_coords,
        valid_combos,
    )
    dir_b = self.circle_direction(
        m_last_coords=self.m_last_coords[0], valid_combos=valid_combos, k=0
    )
    dir_b2 = self.circle_direction(
        m_last_coords=self.m_last_coords[1], valid_combos=valid_combos, k=1
    )
    dir_b3 = self.circle_direction(
        m_last_coords=self.m_last_coords[2], valid_combos=valid_combos, k=2
    )
    self.m_coords_valid = np.array(valid_combos[0])
    if self.is_3d:
        self.plot_mechanism_3d(
            self.p_last_coords,
            self.m_last_coords,
            self.p_coords,
            a,
            a2,
            a3,
            valid_combos=valid_combos,
            axis_range=70,
        )
    self.s_angles_1, self.s_angles_2, self.s_angles_3 = self.s_angles(
        self.s_last_angles[0],
        self.s_last_angles[1],
        self.s_last_angles[2],
        dir_b,
        dir_b2,
        dir_b3,
        a,
        a2,
```

```
            a3,
        )
        status_msg = (
            f"✅ Обрана комбінація: {closest_combo}"
            if closest_combo
            else "⚠️ Немає валідних комбінацій після фільтра"
        )
        if self.is_debug:
            print(
                f"Останні координати платформи:\nC: {self.p_last_coords[0]},\nC2: {self.p_last_coords[1]},\nC3: {self.p_last_coords[2]}\n"
                f"Останні координати middle:\nB: {self.m_last_coords[0]},\nB2: {self.m_last_coords[1]},\nB3: {self.m_last_coords[2]}\n"
                f"Нові координати платформи: {self.p_coords[0]}, {self.p_coords[1]}, {self.p_coords[2]}\n"
                f"{valid_combos}\n"
                f"{status_msg}\n"
                f"{a}, {a2}, {a3}\n"
                f"Знак:\nДля A: {dir_b},\nДля A2: {dir_b2},\nДля A3: {dir_b3}"
                f"Останні кути платформи:\nC: {self.p_last_angles[0]},\nC2: {self.p_last_angles[1]},\nC3: {self.p_last_angles[2]}\n"
                f"Останні кути servo:\nB: {self.s_last_angles[0]},\nB2: {self.s_last_angles[1]},\nB3: {self.s_last_angles[2]}\n"
            )
            print(f"{self.s_angles_1}, {self.s_angles_2}, {self.s_angles_3}")
        self.update_last_coords(
            "platform", self.p_coords[0], self.p_coords[1], self.p_coords[2]
        )
        self.update_last_coords(
            "middle",
            self.m_coords_valid[0],
            self.m_coords_valid[1],
            self.m_coords_valid[2],
        )
        self.update_last_angles("platform", self.alpha, self.beta, self.gamma)

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        self.update_last_angles(
            "servo", self.s_angles_1, self.s_angles_2, self.s_angles_3
        )
        return (self.s_angles_1, self.s_angles_2, self.s_angles_3)
    def p_absolute_angles(
        self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode="delta"
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        if mode == "delta":
            p_absolute_angles_1 = p_last_angles_1 + self.alpha
            p_absolute_angles_2 = p_last_angles_2 + self.beta
            p_absolute_angles_3 = p_last_angles_3 + self.gamma
        elif mode == "absolute":
            p_absolute_angles_1 = self.alpha
            p_absolute_angles_2 = self.beta
            p_absolute_angles_3 = self.gamma
        else:
            raise ValueError("mode має бути 'delta' або 'absolute'")
        p_absolute_angles_1 = (p_absolute_angles_1 + 360) % 360
        p_absolute_angles_2 = (p_absolute_angles_2 + 360) % 360
        p_absolute_angles_3 = (p_absolute_angles_3 + 360) % 360
        return p_absolute_angles_1, p_absolute_angles_2, p_absolute_angles_3
    def p_angles(
        self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode="delta"
    ):
        if mode == "delta":
            p_angles_1 = self.alpha
            p_angles_2 = self.beta
            p_angles_3 = self.gamma
        elif mode == "absolute":
            p_angles_1 = self.alpha - p_last_angles_1
            p_angles_2 = self.beta - p_last_angles_2
            p_angles_3 = self.gamma - p_last_angles_3
        else:
            raise ValueError("mode має бути 'delta' або 'absolute'")
        return p_angles_1, p_angles_2, p_angles_3
    @staticmethod
    def s_angles(
        s_last_angles_1,
        s_last_angles_2,
        s_last_angles_3,
        dir_b,
        dir_b2,
        dir_b3,
        a,
        a2,
        a3,
    ):
        s_angles_1 = s_last_angles_1 + dir_b * a
        s_angles_2 = s_last_angles_2 + dir_b2 * a2
        s_angles_3 = s_last_angles_3 + dir_b3 * a3
        return s_angles_1, s_angles_2, s_angles_3
    def plot_mechanism_3d(
        self,
        p_last_coords,
        m_last_coords,
        p_coords,
        a: float,
        a2: float,
        a3: float,
        *,
```

```
        valid_combos: list | None = None,
        title: str = "3D Visualization",
        axis_range: float = 120,
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        p_last_coords_1, p_last_coords_2, p_last_coords_3 = (
            p_last_coords[0],
            p_last_coords[1],
            p_last_coords[2],
        )
        m_last_coords_1, m_last_coords_2, m_last_coords_3 = (
            m_last_coords[0],
            m_last_coords[1],
            m_last_coords[2],
        )
        p_coords_1, p_coords_2, p_coords_3 = p_coords[0], p_coords[1], p_coords[2]
        m_coords_1, m_coords_2, m_coords_3 = valid_combos[0]
        fig = plt.figure(figsize=(14, 10))
        ax = fig.add_subplot(111, projection="3d")
        style = {
            "platform_old": {"color": "red", "lw": 2},
            "platform_new": {"color": "darkred", "lw": 2, "ls": "--"},
            "link_new": {"color": "green", "lw": 2},
            "link_old": {"color": "blue", "lw": 1, "ls": "--"},
            "link_aux": {"color": "gray", "lw": 1, "ls": "--"},
            "circle_P": {"color": "purple", "ls": "--", "lw": 1.5},
            "circle_O": {"color": "orange", "ls": ":", "lw": 1.5},
            "points": {
                "O": "black",
                "P": "purple",
                "C": "red",
                "C_new": "darkred",
                "B": "blue",
                "B_new": "green",
            },
        }
        def to3d(p: np.ndarray, z: float = 0) -> np.ndarray:
            return np.array([p[0], p[1], z]) if len(p) == 2 else np.array(p)

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        def draw_line(p1: np.ndarray, p2: np.ndarray, style_key="platform") -> None:
            s = style[style_key]
            ax.plot([p1[0], p2[0]], [p1[1], p2[1]], [p1[2], p2[2]], **s)
        def draw_point(p: np.ndarray, label: str, style_key="O") -> None:
            ax.scatter(
                p[0], p[1], p[2], s=50, c=style["points"][style_key], label=label
            )
        def draw_circle(
            center: np.ndarray,
            r: float,
            z_fixed: float | None = None,
            style_key="circle_P",
        ) -> None:
            theta = np.linspace(0, 2 * np.pi, 200)
            x = center[0] + r * np.cos(theta)
            y = center[1] + r * np.sin(theta)
            z = np.full_like(theta, z_fixed if z_fixed is not None else center[2])
            ax.plot(x, y, z, **style[style_key])
        C, C2, C3 = p_last_coords_1, p_last_coords_2, p_last_coords_3
        Cn, C2n, C3n = p_coords_1, p_coords_2, p_coords_3
        B, B2, B3 = to3d(m_last_coords_1), to3d(m_last_coords_2), to3d(m_last_coords_3)
        Bn, B2n, B3n = to3d(m_coords_1), to3d(m_coords_2), to3d(m_coords_3)
        O = np.array([0, 0, 0])
        P = np.array([0, 0, 55])
        for b, c in [(B, C), (B2, C2), (B3, C3)]:
            draw_line(O, b, "link_old")
            draw_line(b, c, "platform_old")
        for b_new, c_new in [(Bn, Cn), (B2n, C2n), (B3n, C3n)]:
            draw_line(O, b_new, "link_aux")
            draw_line(b_new, c_new, "link_new")
        for p1, p2 in [(C, C2), (C2, C3), (C3, C)]:
            draw_line(p1, p2, "platform_old")
        for p1, p2 in [(Cn, C2n), (C2n, C3n), (C3n, Cn)]:
            draw_line(p1, p2, "platform_new")
        for b in [B, B2, B3]:
            draw_line(O, b, "link_old")
        draw_point(O, "O", "O")
        draw_point(P, "P", "P")
        draw_point(C, "C", "C")
        draw_point(C2, "C2", "C")
        draw_point(C3, "C3", "C")
        draw_point(Cn, "C new", "C_new")
        draw_point(C2n, "C2 new", "C_new")
        draw_point(C3n, "C3 new", "C_new")
        draw_point(B, "B", "B")
        draw_point(B2, "B2", "B")
        draw_point(B3, "B3", "B")
        draw_point(Bn, "B new", "B_new")
        draw_point(B2n, "B2 new", "B_new")
        draw_point(B3n, "B3 new", "B_new")
        draw_circle(O, self.R1, z_fixed=O[2], style_key="circle_O")
        draw_circle(P, 40, z_fixed=P[2], style_key="circle_P")
        axis_len = axis_range * 1.1
        ax.quiver(0, 0, 0, axis_len, 0, 0, color="r", arrow_length_ratio=0.1)
        ax.quiver(0, 0, 0, 0, axis_len, 0, color="g", arrow_length_ratio=0.1)
        ax.quiver(0, 0, 0, 0, 0, axis_len, color="b", arrow_length_ratio=0.1)
        ax.text(axis_len, 0, 0, "X", color="r")
        ax.text(0, axis_len, 0, "Y", color="g")
        ax.text(0, 0, axis_len, "Z", color="b")
        text_angles = (
            f"alpha={self.alpha:.1f}°,\n"
            f"beta={self.beta:.1f}°,\n"
```

```
            f"gamma={self.gamma:.1f}°.\n\n"
            f"a={a:.1f}°,\n"
            f"a2={a2:.1f}°,\n"
            f"a3={a3:.1f}°."
        )
        ax.text2D(0.02, 0.95, text_angles, transform=ax.transAxes, fontsize=10)
        ax.set_title(title)
        ax.set_xlabel("X")
        ax.set_ylabel("Y")
        ax.set_zlabel("Z")
        ax.set_box_aspect([1, 1, 1])
        ax.set_xlim(-axis_range, axis_range)
        ax.set_ylim(-axis_range, axis_range)
        ax.set_zlim(0, axis_range)
        plt.legend(loc="center left", bbox_to_anchor=(1.05, 0.5))
        plt.tight_layout()
        plt.show()
    def _get_last_angles(self, angles_type: str) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        mapping = {
            "platform": ["alpha", "beta", "gamma"],
            "servo": ["s1", "s2", "s3"],
        }
        if angles_type not in mapping:
            raise ValueError(
                f"Unknown group '{angles_type}'. Очікується 'platform' або 'servo'."
            )
        keys = mapping[angles_type]
        is_last_flag = getattr(
            self, f"{self.flag_map['angles'][angles_type]}"
        )
        angles_key = "zero" if not is_last_flag else "last"
        values = [self.config.data["angles"][angles_type][angles_key][k] for k in keys]
        if self.is_debug:
            print(f"[DEBUG] values ({angles_type} angles, {angles_key}): {values}")
        if not is_last_flag:
            self.update_last_angles(angles_type, *values)

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            self.config.load()
            setattr(self, f"{self.flag_map['angles'][angles_type]}", True)
        return tuple(values)
    def get_p_last_angles(self) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        return self._get_last_angles("platform")
    def get_s_last_angles(self) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        return self._get_last_angles("servo")

    def circle_direction(self, m_last_coords, valid_combos, k):
        """[пояснення вилучено для скороченого лістингу]"""
        det_x, det_y = self.determ_num(
            m_last_coords=m_last_coords, valid_combos=valid_combos, k=k
        )
        if self.is_debug:
            debug_message = (
                f"Знаки A{" if k == 0 else f'_{k + 1}'}: 'x' - {det_x}, 'y' - {det_y}"
            )
            print(debug_message)
        if det_x == 0 and det_y == 0:
            return 0
        x, y = valid_combos[0][k]
        if x >= 0 and y >= 0:
            return +1 if det_x < 0 or det_y > 0 else -1
        elif x < 0 and y >= 0:
            return +1 if det_x < 0 or det_y < 0 else -1
        elif x < 0 and y < 0:
            return +1 if det_x > 0 or det_y < 0 else -1
        else:
            return +1 if det_x > 0 or det_y > 0 else -1
    @staticmethod
    def choose_closest_combo(combos, last_coords):
        """[пояснення вилучено для скороченого лістингу]"""
        if not combos:
            return None
        def dist(a, b):
            return np.linalg.norm(np.array(a) - np.array(b))
        best_combo = min(
            combos,
            key=lambda combo: sum(dist(c, l) for c, l in zip(combo, last_coords)),
        )
        return best_combo
    def get_p_last_coords(self):
        return self._get_last_coords(kind="platform", recalc=False)
    def get_m_last_coords(self):
        return self._get_last_coords(kind="middle", recalc=False)
    def _get_last_coords(self, *, kind: str, recalc: bool = False):
        """[пояснення вилучено для скороченого лістингу]"""
        mapping = {
            "platform": ["c", "c2", "c3"],
            "middle": ["b", "b2", "b3"],
        }
        if kind not in mapping:
            raise ValueError(
                f"Unknown kind '{kind}'. Очікується 'platform' або 'middle'."
            )
        keys = mapping[kind]
        is_last_flag = getattr(
            self, f"{self.flag_map['coords'][kind]}"
        )
        coords_key = "zero" if not is_last_flag else "last"
        if coords_key == "zero":
            side = "right" if self.is_bend else "left"
```

```
            values = [

self.config.data["coordinates"][kind][coords_key][side]
[k] for k in keys
            ]
        else:
            values = [

self.config.data["coordinates"][kind][coords_key][k]
for k in keys
            ]
        if self.is_debug:
            print(f"[DEBUG] values ({kind}): {values}")
        if (
            recalc
            and kind == "middle"
            and coords_key != "zero"
            and not all(v is not None for v in values)
        ):
            coords = [
                self.get_m_cords(p)
                for p in (
                    self.p_last_coords[0],
                    self.p_last_coords[1],
                    self.p_last_coords[2],
                )
            ]
            if self.is_debug:
                print(f"[DEBUG] recalc coords ({kind}):
{coords}")
        else:
            coords = [np.array(v) for v in values]
            if self.is_debug:
                print(f"[DEBUG] final coords ({kind}):
{coords}")
        if not is_last_flag:
            self.update_last_coords(kind, *coords)
            setattr(self, f"{self.flag_map['coords'][kind]}",
True)
        return tuple(coords)
    @staticmethod
    def segments_intersect(p1, p2, q1, q2, eps_segm=1e-
9):
        """[пояснення вилучено для скороченого
лістингу]"""

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

        def orient(a, b, c):
            return (b[0] - a[0]) * (c[1] - a[1]) - (b[1] - a[1]) *
(c[0] - a[0])
        o1 = orient(p1, p2, q1)
        o2 = orient(p1, p2, q2)
        o3 = orient(q1, q2, p1)
        o4 = orient(q1, q2, p2)
        return o1 * o2 < -eps_segm and o3 * o4 < -
eps_segm
    def is_valid_combo(self, B, B2, B3,
combinations_p):
        """[пояснення вилучено для скороченого
лістингу]"""
        C, C2, C3 = combinations_p
        segments = [(B, C[:2]), (B2, C2[:2]), (B3, C3[:2])]
        for i in range(len(segments)):
            for j in range(i + 1, len(segments)):
                if self.segments_intersect(
                    segments[i][0], segments[i][1],
segments[j][0], segments[j][1]
                ):
                    return False
        return True
    def plot_combinations(self, combinations_m,
combinations_p, decimals=2):
        """[пояснення вилучено для скороченого
лістингу]"""
        if not combinations_m:
            print(" ⚠️ Немає валідних комбінацій")
            return
        C, C2, C3 = combinations_p
        def draw_point(ax, xy, *, color="blue", size=40,
marker="o", label=None, z=3):
            """[пояснення вилучено для скороченого
лістингу]"""
            ax.scatter(
                float(xy[0]),
                float(xy[1]),
                c=color,
                s=size,
                marker=marker,
                zorder=z,
                label=label,
            )
        def draw_line(ax, p1, p2, *, color="k", lw=1, ls="-
", z=2):
            """[пояснення вилучено для скороченого
лістингу]"""
            ax.plot(
                [float(p1[0]), float(p2[0])],
                [float(p1[1]), float(p2[1])],
                linestyle=ls,
                linewidth=lw,
                color=color,
                zorder=z,
            )
        def draw_label(
            ax,
            xy,
            text,
            *,
            fontsize=8,
            color="black",
            ha="left",
            va="bottom",
            offset=(0, 0),
        ):
            """[пояснення вилучено для скороченого
лістингу]"""
            ax.text(
                float(xy[0]) + offset[0],
                float(xy[1]) + offset[1],
                text,
                fontsize=fontsize,
                color=color,
                ha=ha,
                va=va,
            )
```

```
    def xy(p):
        return (float(p[0]), float(p[1]))
    def dist3D(b, c):
        return float(np.linalg.norm([b[0] - c[0], b[1] - c[1], -c[2]]))
    cols = 4
    rows = (len(combinations_m) + cols - 1) // cols
    fig, axes = plt.subplots(rows, cols, figsize=(4 * cols, 4 * rows))
    axes = axes.flatten()
    for ax in axes:
        ax.add_patch(Circle((0, 0), self.R1, fill=False, linestyle="--", alpha=0.6))
        ax.add_patch(Circle((0, 0), 40, fill=False, linestyle="--", alpha=0.6))
    for i, (b, b2, b3) in enumerate(combinations_m):
        ax = axes[i]
        draw_point(ax, b, color="blue")
        draw_point(ax, b2, color="blue")
        draw_point(ax, b3, color="blue")
        draw_point(ax, xy(C), color="red")
        draw_point(ax, xy(C2), color="red")
        draw_point(ax, xy(C3), color="red")
        draw_point(ax, (0, 0), color="black", size=30)
        draw_line(ax, b, b2, color="blue")
        draw_line(ax, b2, b3, color="blue")
        draw_line(ax, b3, b, color="blue")
        draw_line(ax, xy(C), xy(C2), color="red", ls="--")
        draw_line(ax, xy(C2), xy(C3), color="red", ls="--")
        draw_line(ax, xy(C3), xy(C), color="red", ls="--")
        for p in [b, b2, b3]:
            draw_line(ax, (0, 0), p, color="black", ls="--", lw=1)
        # ── лінії: з'єднання пар B–C ──
        draw_line(ax, b, xy(C), color="green", lw=1)
        draw_line(ax, b2, xy(C2), color="green", lw=1)
        draw_line(ax, b3, xy(C3), color="green", lw=1)
        draw_label(

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            ax,
            b,
            f"B({round(b[0], decimals)}; {round(b[1], decimals)})",
            color="red",
            ha="right",
        )
        draw_label(
            ax,
            b2,
            f"B2({round(b2[0], decimals)}; {round(b2[1], decimals)})",
            color="green",
            ha="right",
        )
        draw_label(
            ax,
            b3,
            f"B3({round(b3[0], decimals)}; {round(b3[1], decimals)})",
            color="purple",
            ha="right",
        )
        draw_label(
            ax,
            xy(C),
            f"C({round(C[0], decimals)}; {round(C[1], decimals)})",
            color="red",
        )
        draw_label(
            ax,
            xy(C2),
            f"C2({round(C2[0], decimals)}; {round(C2[1], decimals)})",
            color="red",
        )
        draw_label(
            ax,
            xy(C3),
            f"C3({round(C3[0], decimals)}; {round(C3[1], decimals)})",
            color="red",
        )
        ax.set_title(f"Combo {i + 1}")
        ax.axhline(0, color="black", linewidth=0.5)
        ax.axvline(0, color="black", linewidth=0.5)
        ax.set_aspect("equal", adjustable="box")
    for j in range(len(combinations_m), len(axes)):
        fig.delaxes(axes[j])
    plt.tight_layout()
    plt.show()
  def filter_points_on_circle(self, points, eps_circle=1e-6):
    """[пояснення вилучено для скороченого лістингу]"""
    pts = np.array(points, dtype=float)
    mask = np.abs(np.sum(pts**2, axis=1) - self.R1**2) < eps_circle
    pts = np.round(pts[mask], 8)
    pts = np.unique(pts, axis=0)
    return [tuple(p) for p in pts]
  @staticmethod
  def filter_valid_combinations(
    points, min_dist=10.0, eps_abs=1e-9, round_decimals=8
  ):
    """[пояснення вилучено для скороченого лістингу]"""
    B, B2, B3 = points
    def _unique_points(points):
        seen = set()
        out = []
        for x, y in points:
            key = (round(float(x), round_decimals), round(float(y), round_decimals))
            if key not in seen:
                seen.add(key)
                out.append(key)
        return out
    B = _unique_points(B)
    B2 = _unique_points(B2)
    B3 = _unique_points(B3)
```

```
        md2 = float(min_dist) ** 2
        eps2 = float(eps_abs) ** 2
        thr2 = max(md2, eps2)
        valid_combinations = []
        seen_combos = set()
        for combo in itertools.product(B, B2, B3):
            ok = True
            for (x1, y1), (x2, y2) in
itertools.combinations(combo, 2):
                dx, dy = x2 - x1, y2 - y1
                if dx * dx + dy * dy <= thr2:
                    ok = False
                    break
            if not ok:
                continue
            key = tuple(
                (round(x, round_decimals), round(y,
round_decimals)) for (x, y) in combo
            )
            if key in seen_combos:
                continue
            seen_combos.add(key)
            valid_combinations.append(tuple((float(x),
float(y)) for (x, y) in combo))
        return valid_combinations
    def get_config(self) -> tuple[dict, dict, dict, dict]:
        """[пояснення вилучено для скороченого
лістингу]"""
        self.config                                    =
AnglesConfig(work_dirs=self.work_dirs,
is_multi=self.is_multi)
        if self.is_debug:
            print(f"[DEBUG] Config loaded:
{self.config.config_path}")
        self.all_p_coors                               =
self.config.data["coordinates"]["platform"]

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

        self.all_m_coors                               =
self.config.data["coordinates"]["middle"]
        self.all_p_angles                              =
self.config.data["angles"]["platform"]
        self.all_s_angles                              =
self.config.data["angles"]["servo"]
        return self.all_p_coors, self.all_m_coors,
self.all_p_angles, self.all_s_angles
    def update_last_coords(self, coords_type, coords_1,
coords_2, coords_3):
        """[пояснення вилучено для скороченого
лістингу]"""
        mapping = {
            "platform": ["c", "c2", "c3"],
            "middle": ["b", "b2", "b3"],
        }
        if coords_type not in mapping:
            raise ValueError(f"Unknown coord_type
'{coords_type}'")
        keys = mapping[coords_type]
        last_data = {
            keys[0]: coords_1.tolist(),
            keys[1]: coords_2.tolist(),
            keys[2]: coords_3.tolist(),
        }
        all_entry = [coords_1.tolist(), coords_2.tolist(),
coords_3.tolist()]

self.config.update_value(f"coordinates.{coords_type}.l
ast", last_data)
        cfg = self.config.data

cfg["coordinates"][coords_type]["all"].append(all_entr
y)
        self.config.update_value(
            f"coordinates.{coords_type}.all",
            cfg["coordinates"][coords_type]["all"],
        )
        self.config.load()
    def update_last_angles(
        self,
        angles_type: str,
        angles_1: float,
        angles_2: float,
        angles_3: float,
    ) -> None:
        """[пояснення вилучено для скороченого
лістингу]"""
        mapping = {
            "platform": ["alpha", "beta", "gamma"],
            "servo": ["a", "a2", "a3"],
        }
        if angles_type not in mapping:
            raise ValueError(
                f"Unknown group '{angles_type}'.
Очікується 'platform' або 'servo'."
            )
        keys = mapping[angles_type]
        last_data = {
            keys[0]: float(angles_1),
            keys[1]: float(angles_2),
            keys[2]: float(angles_3),
        }
        all_entry = [float(angles_1), float(angles_2),
float(angles_3)]

self.config.update_value(f"angles.{angles_type}.last",
last_data)
        cfg = self.config.data

cfg["angles"][angles_type]["all"].append(all_entry)
        self.config.update_value(
            f"angles.{angles_type}.all",
            cfg["angles"][angles_type]["all"],
        )
        self.config.load()
        setattr(self, f"is_{angles_type[0]}_last_angles",
True)
    def get_p_coords(self, coords):
        return self._rotation_matrix() @ coords
    def _rotation_matrix(self):
        a, b, g = np.radians([self.alpha, self.beta,
self.gamma])
        ca, cb, cg = np.cos([a, b, g])
        sa, sb, sg = np.sin([a, b, g])
        return np.array(
            [
```

```
            [cb * cg, cg * sa * sb - ca * sg, sa * sg + ca * cg * sb],
            [cb * sg, ca * cg + sa * sb * sg, ca * sb * sg - cg * sa],
            [sb, -cb * sa, ca * cb],
        ]
    )
  def get_m_cords(self, m):
    x, y, z = float(m[0]), float(m[1]), float(m[2])
    if z * z > self.Rc * self.Rc:
      if self.is_debug:
        print(
          f"⚠️ Неприпустиме z: z^2={z * z:.6f} > Rc^2={self.Rc * self.Rc:.6f}"
        )
      return None
    r2 = np.sqrt(self.Rc**2 - z**2)
    A = -(self.R1**2) + 2 * self.R1 * r2 - r2**2 + x**2 + y**2
    B = self.R1**2 + 2 * self.R1 * r2 + r2**2 - x**2 - y**2
    under_sqrt = A * B
    if under_sqrt < 0:
      if self.is_debug:
        print(
          "⚠️ Уявні координати (підкореневий вираз < 0). "
          f"alpha={self.alpha:.6f}, beta={self.beta:.6f}, gamma={self.gamma:.6f}; "
          f"m=({x:.6f},{y:.6f},{z:.6f}); under_sqrt={under_sqrt:.6e}"
        )
      return None
    f = np.sqrt(under_sqrt)
    denom1 = 2 * x * (x**2 + y**2)
    denom2 = 2 * (x**2 + y**2)
    tiny = 1e-12
    if abs(denom1) < tiny or abs(denom2) < tiny:
      if self.is_debug:
        print(

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

          "⚠️ Вироджена геометрія (ділення дуже мале). "
          f"denom1={denom1:.6e}, denom2={denom2:.6e}, m=({x},{y})"
        )
      return None
    bx1 = (
      y * (x * f - y * (self.R1**2 - r2**2 + x**2 + y**2))
      + (x**2 + y**2) * (self.R1**2 - r2**2 + x**2 + y**2)
    ) / denom1
    bx2 = (
      -y * (x * f + y * (self.R1**2 - r2**2 + x**2 + y**2))
      + (x**2 + y**2) * (self.R1**2 - r2**2 + x**2 + y**2)
    ) / denom1
    by1 = (-x * f + y * (self.R1**2 - r2**2 + x**2 + y**2)) / denom2
    by2 = (x * f + y * (self.R1**2 - r2**2 + x**2 + y**2)) / denom2
    return float(bx1), float(bx2), float(by1), float(by2)
  def filter_by_bend_side(self, combos, combinations_p):
    """[пояснення вилучено для скороченого лістингу]"""
    if not combos:
      return []
    C, C2, C3 = combinations_p
    def vector_angle_sign(C, B):
      """[пояснення вилучено для скороченого лістингу]"""
      vC = np.array([C[0], C[1]])
      vB = np.array([B[0], B[1]])
      cross = vC[0] * vB[1] - vC[1] * vB[0]
      return np.sign(cross)
    def combo_bend_sign(combo):
      """[пояснення вилучено для скороченого лістингу]"""
      (B, B2, B3) = combo
      signs = [
        vector_angle_sign(C[:2], B),
        vector_angle_sign(C2[:2], B2),
        vector_angle_sign(C3[:2], B3),
      ]
      return np.sign(sum(signs))
    desired_sign = -1 if self.is_bend else +1
    filtered = [combo for combo in combos if combo_bend_sign(combo) == desired_sign]
    if not filtered and self.is_debug:
      print(
        f"⚠️ Всі {len(combos)} комбінацій не збігаються з is_bend={self.is_bend}. "
        f"Повертаємо весь набір без фільтрації."
      )
      return combos
    return filtered
  def get_angles(
    self,
    m_last_coords,
    valid_combos: list[
      tuple[tuple[float, float], tuple[float, float], tuple[float, float]]
    ],
  ) -> tuple[float, float, float]:
    """[пояснення вилучено для скороченого лістингу]"""
    m_last_coords_1, m_last_coords_2, m_last_coords_3 = m_last_coords
    b_vectors = MathFunction.get_vector(
      m_last_coords_1[0],
      m_last_coords_1[1],
      valid_combos[0][0][0],
      valid_combos[0][0][1],
    )
    b2_vectors = MathFunction.get_vector(
      m_last_coords_2[0],
      m_last_coords_2[1],
      valid_combos[0][1][0],
      valid_combos[0][1][1],
    )
```

```
        b3_vectors = MathFunction.get_vector(
            m_last_coords_3[0],
            m_last_coords_3[1],
            valid_combos[0][2][0],
            valid_combos[0][2][1],
        )
        a_sin = MathFunction.sin_alpha(b_vectors, self.R1)
        a2_sin = MathFunction.sin_alpha(b2_vectors, self.R1)
        a3_sin = MathFunction.sin_alpha(b3_vectors, self.R1)
        if self.is_debug:
            print(
                f"{self.m_last_coords[0]}, {m_last_coords_1[1]}, {valid_combos[0][0][0]}, {valid_combos[0][0][1]}\n"
                f"Вектор для точки B: {b_vectors}"
            )
            print(
                f"{m_last_coords_2[0]}, {m_last_coords_2[1]}, {valid_combos[0][1][0]}, {valid_combos[0][1][1]}\n"
                f"Вектор для точки B2: {b2_vectors}"
            )
            print(
                f"{m_last_coords_3[0]}, {m_last_coords_3[1]}, {valid_combos[0][2][0]}, {valid_combos[0][2][1]}\n"
                f"Вектор для точки B3: {b3_vectors}"
            )
            print(f"Синуси кутів: A - {a_sin}, A2 - {a2_sin}, A3 - {a3_sin}")
        a = 180 / np.pi * arcsin(a_sin)
        a2 = 180 / np.pi * arcsin(a2_sin)
        a3 = 180 / np.pi * arcsin(a3_sin)
        return a, a2, a3
    @staticmethod
    def determ_num(
        *,
        m_last_coords: tuple[float, float],
        valid_combos: list[
            tuple[tuple[float, float], tuple[float, float], tuple[float, float]]
        ],

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        k: int,
    ) -> tuple[int, int]:
        """[пояснення вилучено для скороченого лістингу]"""
        det_x = valid_combos[0][k][0] - m_last_coords[0]
        det_y = valid_combos[0][k][1] - m_last_coords[1]
        sign = lambda v: int(v > 0) - int(v < 0)
        return sign(det_x), sign(det_y)
    @staticmethod
    def build_parser():
        epilog_text = """[пояснення вилучено для скороченого лістингу]"""
        parser = argparse.ArgumentParser(
            prog="get_angles",
            description="Обчислення кутів сервоприводів (A, A2, A3) для заданих α, β, γ.",

formatter_class=argparse.RawDescriptionHelpFormatter,
            epilog=epilog_text,
        )
        parser.add_argument("alpha", type=float, help="Кут α (в градусах)")
        parser.add_argument("beta", type=float, help="Кут β (в градусах)")
        parser.add_argument("gamma", type=float, help="Кут γ (в градусах)")
        parser.add_argument(
            "--is_bend",
            action=argparse.BooleanOptionalAction,
            default=True,
            help="Вибір сторони згину: True — права, False — ліва",
        )
        parser.add_argument(
            "--is_debug",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="[DEBUG_INFO] Режим налагодження (додаткові повідомлення). (default: False)",
        )
        parser.add_argument(
            "--is_multi",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Багатократний режим (для пакетної обробки). (default: False)",
        )
        parser.add_argument(
            "--rc",
            type=float,
            default=75.0,
            help="Радіус Rc",
        )
        parser.add_argument(
            "--r1",
            type=float,
            default=50.0,
            help="Радіус R1",
        )
        parser.add_argument(
            "--is_2d",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Побудова 2D графіків",
        )
        parser.add_argument(
            "--is_3d",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Побудова 3D графіків",
        )
        return parser
    def run_terminal(self):
        parser = self.build_parser()
        if len(sys.argv) == 1:
            parser.print_help()
            sys.exit(0)
        args = parser.parse_args()
```

```
        app = GetAngles(
            is_bend=args.is_bend,
            is_debug=args.is_debug,
            is_multi=args.is_multi,
            rc=args.rc,
            r1=args.r1,
            is_2d=args.is_2d,
            is_3d=args.is_3d,
        )
        result = app.calculation(alpha=args.alpha,
beta=args.beta, gamma=args.gamma)
        print(result)
        return result
    def run(self):
        result = self.calculation(
            alpha=self.alpha,
            beta=self.beta,
            gamma=self.gamma,
        )
        if self.is_debug:
            print(f"[OK] Run finished. Servo angles:
{result}")
        return result
if __name__ == "__main__":
    app = GetAngles()
    if len(sys.argv) > 1:
        app.run_terminal()
    else:
        print("[INFO] Running GetAngles in default mode
(no CLI args).")
        app.run()

#
=======================================
=========
# === objtoangles/objtoangles.py
#
=======================================
=========

"""[пояснення вилучено для скороченого
лістингу]"""
import os
import argparse
import json
import sys
import numpy as np
import trimesh
import matplotlib.pyplot as plt
from pathlib import Path
from common.project_manager import
get_default_work_dirs
class ObjToAngles:
    def __init__(
        self,
        input_file: str | None = None,
        output_file: str | None = None,
        filtered_file: str | None = None,
        *,
        unit: str = "deg",
        unique: bool = True,
        sample: int = 0,
        line_samples: int = 50,
        is_filter: bool = False,
        is_clear: bool = False,
        is_debug: bool = False,
    ):
        """[пояснення вилучено для скороченого
лістингу]"""
        self.work_dirs = get_default_work_dirs()
        self.obj_dir = self.work_dirs.obj_dir
        self.json_dir = self.work_dirs.json_dir
        defaults = {
            "input": ("input.obj", self.obj_dir),
            "output": ("output.json", self.json_dir),
            "filtered": ("filtered.json", self.json_dir),
        }
        def resolve_path(filename: str | None, key: str):
            default_name, base_dir = defaults[key]
            base_dir = Path(base_dir)
            if filename:
                return filename if os.path.isabs(filename) else
base_dir / filename
            return base_dir / default_name
        self.input_file = resolve_path(input_file, "input")
        self.output_file = resolve_path(output_file,
"output")
        self.filtered_file = resolve_path(filtered_file,
"filtered")
        self.unit = unit
        self.unique = unique
        self.sample = sample
        self.line_samples = line_samples
        self.is_filter = is_filter
        self.is_clear = is_clear
        self.is_debug = is_debug
        self.selected_indices = set()
    @staticmethod
    def compute_angles(point, unit="deg"):
        x, y, z = point
        alpha = np.arctan2(y, z)
        beta = np.arctan2(x, z)
        gamma = np.arctan2(y, x)
        if unit == "deg":
            alpha = np.degrees(alpha)
            beta = np.degrees(beta)
            gamma = np.degrees(gamma)
        return float(alpha), float(beta), float(gamma)
    @staticmethod
    def unique_rows_preserve_order(arr):
        seen = {}
        unique = []
        for row in arr:
            key = tuple(map(float, row))
            if key not in seen:
                seen[key] = True
                unique.append(row)
        return np.array(unique, dtype=float)
    @staticmethod
    def discretize_lines(mesh, line_samples=50):
        points = []
        if hasattr(mesh, "entities"):
            for ent in mesh.entities:
                if
ent.__class__.__name__.lower().startswith("line"):
                    verts = mesh.vertices[ent.points]
                    for i in range(len(verts) - 1):
                        p0, p1 = verts[i], verts[i + 1]
                        t = np.linspace(0, 1, line_samples)
                        seg = np.outer(1 - t, p0) + np.outer(t, p1)
```

```
                points.extend(seg)
        return np.array(points)
    def visualize_and_select_from_file(self, input_json, output_json="filtered.json"):
        with open(input_json, "r", encoding="utf-8") as f:
            data = json.load(f)
        points = data["points"]
        angles = data["angles"]
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        xs = [p["x"] for p in points]
        ys = [p["y"] for p in points]
        zs = [p["z"] for p in points]
        ax.scatter(xs, ys, zs, c="blue", picker=True, s=30)
        def on_pick(event):
            ind = event.ind[0]
            if ind in self.selected_indices:
                self.selected_indices.remove(ind)


raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

                ax.scatter(xs[ind], ys[ind], zs[ind], c="blue", s=50)
            else:
                self.selected_indices.add(ind)
                ax.scatter(xs[ind], ys[ind], zs[ind], c="red", s=80, marker="o")
            plt.draw()
        def on_key(event):
            if event.key.lower() in {"e", "y"}:
                filtered_points = [points[i] for i in self.selected_indices]
                filtered_angles = [angles[i] for i in self.selected_indices]
                output_data = {"points": filtered_points, "angles": filtered_angles}
                with open(output_json, "w", encoding="utf-8") as f_out:
                    json.dump(
                        output_data,
                        f_out,
                        ensure_ascii=False,
                        indent=2,
                    )
                print(f"[OK] Saved {len(filtered_points)} points -> {output_json}")
        fig.canvas.mpl_connect("pick_event", on_pick)
        fig.canvas.mpl_connect("key_press_event", on_key)
        plt.title("Клік по точках = вибір, 'E' = зберегти (export)")
        plt.show()
    def process_mesh(
        self,
        input_file: str | None = None,
        output_file: str | None = None,
        filtered_file: str | None = None,
        *,
        unit: str | None = None,
        unique: bool | None = None,
        sample: int | None = None,
        line_samples: int | None = None,
        is_filter: bool | None = None,
        is_clear: bool | None = None,
        is_debug: bool | None = None,
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        input_file = input_file or self.input_file
        output_file = output_file or self.output_file
        filtered_file = filtered_file or self.filtered_file
        unit = unit or self.unit
        unique = self.unique if unique is None else unique
        sample = self.sample if sample is None else sample
        line_samples = self.line_samples if line_samples is None else line_samples
        is_filter = self.is_filter if is_filter is None else is_filter
        is_clear = self.is_clear if is_clear is None else is_clear
        is_debug = self.is_debug if is_debug is None else is_debug
        scene_or_mesh = trimesh.load(input_file)
        if isinstance(scene_or_mesh, trimesh.Scene):
            meshes = list(scene_or_mesh.geometry.values())
            if not meshes and is_debug:
                print("❌ У сцені немає геометрії")
                sys.exit(2)
            mesh = trimesh.util.concatenate(meshes)
        else:
            mesh = scene_or_mesh
        verts = np.asarray(mesh.vertices)
        line_points = self.discretize_lines(mesh, line_samples)
        if line_points.size > 0:
            verts = np.vstack((verts, line_points))
        if verts.size == 0 and is_debug:
            print("❌ В obj немає вершин або ліній.")
            sys.exit(2)
        if unique:
            verts = self.unique_rows_preserve_order(verts)
        if sample and 0 < sample < len(verts):
            rng = np.random.default_rng(0)
            idx = rng.choice(len(verts), size=sample, replace=False)
            verts = verts[np.sort(idx)]
        result = {"points": [], "angles": []}
        for v in verts:
            alpha, beta, gamma = self.compute_angles(v, unit=unit)
            x, y, z = v
            y, z = z, y
            result["points"].append({"x": x, "y": y, "z": z})
            result["angles"].append({"alpha": alpha, "beta": beta, "gamma": gamma})
        with open(output_file, "w", encoding="utf-8") as f:
            json.dump(result, f, ensure_ascii=False, indent=2)
        if is_debug:
            print(f"[OK] Saved {len(result['points'])} points → {output_file}")
        if is_filter:
            self.visualize_and_select_from_file(output_file, filtered_file)
        if is_clear and os.path.exists(output_file):
```

```
        os.remove(output_file)
    @staticmethod
    def build_parser():
        epilog_text = """[пояснення вилучено для скороченого лістингу]"""
        parser = argparse.ArgumentParser(
            prog="obj_to_angles",
            description="Convert OBJ → JSON with calculated angles (α, β, γ) for each vertex or line point.",

formatter_class=argparse.RawDescriptionHelpFormatter,
            epilog=epilog_text,
        )
        parser.add_argument("input", help="input OBJ file")
        parser.add_argument("output", help="output JSON file")
        parser.add_argument(
            "--unit",
            choices=["deg", "rad"],
            default="deg",

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            help="units for angles (default: deg)",
        )
        parser.add_argument(
            "--unique-vertices",
            dest="unique",
            action=argparse.BooleanOptionalAction,
            default=True,
            help="remove duplicate vertices (default: True) or keep all vertices, even duplicates (False)",
        )
        parser.add_argument(
            "--sample",
            type=int,
            default=0,
            help="random sample of N points (0 = all, default: 0)",
        )
        parser.add_argument(
            "--line-samples",
            type=int,
            default=50,
            help="number of points per line segment (default: 50)",
        )
        parser.add_argument(
            "--filter",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="enable the point sampling function (saves to filtered.json, default: False)",
        )
        parser.add_argument(
            "--clear",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="delete intermediate file 'output.json' (default: False)",
        )
        parser.add_argument(
            "--is_debug",
            action=argparse.BooleanOptionalAction,
            default=False,
            help=" [DEBUG_INFO] (default: False)",
        )
        return parser
    def run_terminal(self):
        parser = self.build_parser()
        if len(sys.argv) == 1:
            parser.print_help()
            sys.exit(0)
        args = parser.parse_args()
        self.process_mesh(
            args.input,
            args.output,
            unit=args.unit,
            unique=args.unique,
            sample=args.sample,
            line_samples=args.line_samples,
            is_filter=args.filter,
            is_clear=args.clear,
            is_debug=args.is_debug,
        )
    def run(self):
        self.process_mesh(
            self.input_file,
            self.output_file,
            self.filtered_file,
            unit=self.unit,
            unique=self.unique,
            sample=self.sample,
            line_samples=self.line_samples,
            is_filter=self.is_filter,
            is_clear=self.is_clear,
            is_debug=self.is_debug,
        )
if __name__ == "__main__":
    app = ObjToAngles()
    if len(sys.argv) > 1:
        app.run_terminal()
    else:
        print("[INFO] Running ObjToAngles in default mode (no CLI args).")
        app.run()

#
=========================================
=========
# === project_listing_protected.py
#
=========================================
=========

import threading
import subprocess
import sys, os, runpy, json, shutil
from pathlib import Path
from common.project_manager import WorkDirs, ConfigManager
class ApiConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.3
```

```
    def __init__(
        self,
        config_name: str = "api_config.json",
        work_dirs=None,
        template_name: str = "api_config.json.default",
    ):
        self.work_dirs = work_dirs
        self.template_name = template_name
        base_dir = work_dirs.json_dir if work_dirs else "json"
        self.base_dir = base_dir
        self.config_path = Path(base_dir) / config_name
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.template_path = Path(base_dir) / template_name
        if not self.template_path.exists():
            self._save_default_config_template()
        self.ensure_config()

super().__init__(config_name=self.config_path.name, base_dir=self.base_dir)
        self.load()
    def _save_default_config_template(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_config = self._get_default_structure()
        with open(self.template_path, "w", encoding="utf-8") as f:
            json.dump(default_config, f, ensure_ascii=False, indent=4)
        print(f"[OK] Created default API template: {self.template_path}")
    def ensure_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        if self.config_path.exists():
            return
        if not self.template_path.exists():
            self._save_default_config_template()
        try:
            shutil.copyfile(self.template_path, self.config_path)
            with open(self.config_path, "r", encoding="utf-8") as f:
                self.data = json.load(f)
        except (json.JSONDecodeError, FileNotFoundError, OSError) as e:
            if self.work_dirs and self.work_dirs.is_debug:
                print(
                    f"[WARNING] template copy/load failed ({e}), writing default structure"
                )
            default_config = self._get_default_structure()
            os.makedirs(self.config_path.parent, exist_ok=True)
            with open(self.config_path, "w", encoding="utf-8") as f:
                json.dump(default_config, f, ensure_ascii=False, indent=4)
            self.data = default_config
            if self.work_dirs and self.work_dirs.is_debug:
                print(f"[DEBUG] Created config: {self.config_path}")
    def validate_structure(self):
        required_keys = {"server", "paths", "modules", "ui", "debug"}
        if not isinstance(self.data, dict):
            return False
        return required_keys.issubset(self.data.keys())
    def load(self):
        super().load()
        if not self.validate_structure():
            print(f"[WARNING] Invalid API config structure → reset to default")
            self.ensure_config()
            super().load()
        elif self.data.get("version", 0) < self.CURRENT_VERSION:
            print(f"[INFO] Updating API config version")
            self.data["version"] = self.CURRENT_VERSION
            self.save()
        return self.data
    def update_value(self, path_str: str, new_value, autosave: bool = True):
        """[пояснення вилучено для скороченого лістингу]"""
        keys = path_str.split(".")
        d = self.data
        for k in keys[:-1]:
            d = d[k]
        d[keys[-1]] = new_value
        if autosave:
            self.save()
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        return {
            "version": self.CURRENT_VERSION,
            "server": {
                "_comment": "⚙️ Параметри локального FastAPI сервера",
                "host": "0.0.0.0",
                "port": 8765,
                "log_level": "info",
                "auto_restart": True,
            },
            "paths": {
                "_comment": "📁 Шляхи до основних робочих і допоміжних тек",
                "default_work_dir": "",
                "modules_root": "./",
                "temp_dir": "./temp",
                "logs_dir": "./logs",
                "json_dir": "./json",
                "obj_dir": "./obj",
            },

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
            "extensions": {
                "_comment": "🧱 Дозволені типи файлів для завантаження /upload",
```

```
            "allowed": [".obj", ".json"],
            "max_size_mb": 50,
        },
        "modules": {
            "_comment": "🧩 Опис активних модулів системи (можна вмикати/вимикати окремо)",
            "determ_work_area": {
                "_comment": "Розрахунок робочої області",
                "entry": "determ_work_area/determ_work_area.py",
                "enabled": True,
                "default_args": {
                    "input_obj": "",
                    "conv_out": "",
                    "valid_out": "",
                    "valid_res": "",
                    "output": "",
                    "unoutput": "",
                    "gen_new_path": False,
                    "is_new_path": False,
                    "is_validate": True,
                    "is_simpify": False,
                    "is_check": True,
                    "is_debug": False,
                },
            },
            "getangles": {
                "_comment": "Розрахунок кутів на основі точок або об’єктів",
                "entry": "getangles/getangles.py",
                "enabled": True,
                "default_args": {
                    "is_bend": True,
                    "is_debug": False,
                    "is_multi": False,
                    "rc": 75.0,
                    "r1": 50.0,
                    "is_2d": False,
                    "is_3d": False,
                },
            },
            "objtoangles": {
                "_comment": "Перетворення OBJ моделей у кути або точки",
                "entry": "objtoangles/objtoangles.py",
                "enabled": True,
                "default_args": {
                    "input": "",
                    "output": "",
                    "--unit": "rad",
                    "--unique-vertices": "unique",
                    "--sample": 0,
                    "--line-samples": 50,
                    "--filter": False,
                    "--clear": False,
                    "--is_debug": False,
                },
            },
            "send_to_arduino": {
                "_comment": "Передача даних у пристрій Arduino через COM-порт",
                "entry": "send_to_arduino/to_arduino.py",
                "enabled": False,
            },
            "ai_rtt": {
                "_comment": "Модуль управління платформою через ComputerVision (AIVision) у режимі реального часу (RTT)",
                "entry": "rtt_controllers/ai_rtt.py",
                "enabled": False,
            },
            "joystick": {
                "_comment": "Модуль управління платформою через Joystick у режимі реального часу (RTT)",
                "entry": "rtt_controllers/joystick.py",
                "enabled": False,
            },
        },
        "visualization": {
            "_comment": "🔍 Керування візуалізацією у віддаленому режимі",
            "disable_remote": True,
            "force_allow": False,
            "flags": ["is_simpify", "is_2d", "is_3d", "--filter"],
        },
        "ui": {
            "_comment": "🖥️ Параметри графічного інтерфейсу користувача (GUI)",
            "auto_start": True,
            "check_server_before_launch": True,
        },
        "debug": {
            "_comment": "🧠 Налагодження і логування",
            "is_debug": True,
            "log_to_file": True,
            "log_file": "server.log",
        },
        "ip": {"allowed": [], "disallowed": []},
    }
  def server(self):
    return self.data.get("server", {})
  def paths(self):
    return self.data.get("paths", {})
  def extensions(self):
    return self.data.get("extensions")
  def modules(self):

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

    return self.data.get("modules", {})
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
  def visualization(self):
    return self.data.get("visualization", {})
  def debug(self):
    return self.data.get("debug", {})
  def ui(self):
    return self.data.get("ui", {})
  def work_with_ip(self):
    return self.data.get("ip", {})
class ProcessManager:
```

```
    """[пояснення вилучено для скороченого лістингу]"""
    def __init__(self, work_dirs: WorkDirs | None = None):
        self.work_dirs = work_dirs
        self.config = ApiConfig(work_dirs=self.work_dirs)
        self.server_config = self.config.server()
        self.paths_config = self.config.paths()
        self.extension_config = self.config.extensions()
        self.modules_config = self.config.modules()
        self.visualization_config = self.config.visualization()
        self.debug_config = self.config.debug()
        self.ui_config = self.config.ui()
        self.work_with_ip = self.config.work_with_ip()
        self.active_processes = {}
        self.is_debug = self.debug_config["is_debug"]
    def _get_my_work_dir(self):
        return self.work_dirs.__repr__()
    def start_process(self, name: str, command: list[str]) -> None:
        """[пояснення вилучено для скороченого лістингу]"""
        if self.is_debug:
            print(f"[DEBUG] Starting process '{name}' → {' '.join(command)}")
        env = os.environ.copy()
        if self.work_dirs and hasattr(self.work_dirs, "base_dir"):
            env["PROJECT_ROOT"] = str(self.work_dirs.base_dir)
        env["SPM_SERVER_CHILD"] = "1"
        process = subprocess.Popen(
            command,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True,
            env=env,
        )
        self.active_processes[name] = process
        if self.is_debug:
            def _stream_output(stream, prefix):
                for line in iter(stream.readline, ""):
                    print(f"[{name}][{prefix}] {line.strip()}")
                stream.close()
            threading.Thread(
                target=_stream_output, args=(process.stdout, "OUT"), daemon=True
            ).start()
            threading.Thread(
                target=_stream_output, args=(process.stderr, "ERR"), daemon=True
            ).start()
        log_path = Path(self.work_dirs.base_dir) / f"logs/{name}.log"
        log_path.parent.mkdir(exist_ok=True)
        log_file = open(log_path, "w", encoding="utf-8")
        process = subprocess.Popen(
            command,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True,
            env=env,
        )
        def _stream_output(stream, prefix):
            for line in iter(stream.readline, ""):
                print(f"[{name}][{prefix}] {line.strip()}")
                log_file.write(f"[{prefix}] {line}")
                log_file.flush()
            stream.close()
        threading.Thread(
            target=_stream_output, args=(process.stdout, "OUT"), daemon=True
        ).start()
        threading.Thread(
            target=_stream_output, args=(process.stderr, "ERR"), daemon=True
        ).start()
        if self.is_debug:
            print(f"[DEBUG] PID {process.pid} started for {name}")
    def check_status(self, name: str) -> dict[str, str | None] | str:
        """[пояснення вилучено для скороченого лістингу]"""
        process = self.active_processes.get(name)
        if not process:
            return {"status": "not_found", "exit_code": None}
        retcode = process.poll()
        if retcode is None:
            return {"status": "running", "exit_code": None}
        stderr = ""
        try:
            if process.stderr:
                stderr = process.stderr.read().strip()
        except Exception:
            pass
        status = "finished_ok" if retcode == 0 else "error"
        return {
            "status": status,
            "exit_code": retcode,
            "stderr": stderr,
        }
    def stop_process(self, name: str) -> None:

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        """[пояснення вилучено для скороченого лістингу]"""
        process = self.active_processes.get(name)
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
        if not process:
            print(f"[WARNING] Process '{name}' not found.")
            return
        process.terminate()
        process.wait(timeout=5)
        if self.is_debug:
            print(f"[DEBUG] Process '{name}' terminated.")
    def run_module(self, name: str, args: dict | None = None):
        """[пояснення вилучено для скороченого лістингу]"""
```

```
        module = self.modules_config.get(name)
        if not module:
            print(f"[ERROR] Module '{name}' not found in
api_config.json.")
            return
        if not module.get("enabled", True):
            print(f"[INFO] Module '{name}' is disabled in
api_config.json.")
            return
        entry = module.get("entry")
        if not entry:
            print(f"[ERROR] Module '{name}' has no entry
path defined.")
            return
        candidate = Path(self.work_dirs.base_dir) / entry
        internal_candidate = Path(self.work_dirs.base_dir)
/ "_internal" / entry
        script_path = candidate if candidate.exists() else
internal_candidate
        if not script_path.exists():
            print(f"[ERROR] Script {script_path} not
found.")
            return
        vis_cfg = self.visualization_config or {}
        disable_remote = vis_cfg.get("disable_remote",
True)
        force_allow = vis_cfg.get("force_allow", False)
        vis_flags = vis_cfg.get("flags", [])
        use_defaults = bool(args and
args.get("default_args", False))
        if use_defaults:
            args = module.get("default_args", {}).copy()
            print(f"[INFO] Використовуються default_args
для '{name}' → {args}")
        elif args:
            args.pop("default_args", None)
        else:
            args = {}
        is_remote = (
            not sys.stdin.isatty()
            or os.environ.get("SSH_CONNECTION")
            or os.environ.get("REMOTE_CONSOLE")
        )
        if disable_remote and not force_allow and
is_remote:
            print("[WARN] Візуалізація недоступна для
віддаленого запуску.")
            for k in list(args.keys()):
                if any(
                    flag.lower().lstrip("-")
k.lower().lstrip("-")
                    for flag in vis_flags
                ):
                    args[k] = False
                args["_warning"] = "Візуалізація
недоступна для віддаленого запуску."
        cli_args = []
        for k, v in args.items():
            if k.startswith("_"):
                continue
            if k in ("input", "output", "alpha", "beta",
"gamma"):
                cli_args.append(str(v))
            elif isinstance(v, bool):
                cli_args.append(f"--{k}" if v else f"--no-{k}")
            else:
                cli_args.extend([f"--{k}", str(v)])
        if getattr(sys, "frozen", False):
            print(f"[INFO] ▶ Виконуємо '{name}' inline
(всередині поточного процесу).")
            argv_backup = sys.argv.copy()
            sys.argv = [str(script_path)] + cli_args
            script_dir = os.path.dirname(str(script_path))
            if script_dir not in sys.path:
                sys.path.insert(0, script_dir)
            try:
                runpy.run_path(str(script_path),
run_name="__main__")
            finally:
                sys.argv = argv_backup
            return
        cmd = [sys.executable, str(script_path)] + cli_args
        self.start_process(name, cmd)
    def run_all_enabled_modules(self):
        for name, module in self.modules_config.items():
            if module.get("enabled", False):
                self.run_module(name)
    def cleanup(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        for name, process in self.active_processes.items():
            if process.poll() is None:
                process.terminate()
                if self.is_debug:
                    print(f"[DEBUG] Terminated process
'{name}' (PID {process.pid}).")
    def shutdown(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        for name, process in self.active_processes.items():
            if process.poll() is None:
                print(f"Stopping {name}...")
                process.terminate()
                try:
                    process.wait(timeout=5)
                except subprocess.TimeoutExpired:


raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---


                    print(f"[FORCE] Killing {name}...")
                    process.kill()
    def get_status_report(self) -> dict:
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
        report = {}
        for name, process in self.active_processes.items():
            status = "running" if process.poll() is None else
"finished"
            report[name] = {"pid": process.pid, "status":
status}
        return report
    def check_ip_access(self, request) -> bool:
        """[пояснення вилучено для скороченого
лістингу]"""
        client_ip = None
        try:
```

```
        client_ip = request.client.host if request.client else None
    except Exception:
        pass
    allowed = self.work_with_ip.get("allowed", [])
    disallowed = self.work_with_ip.get("disallowed", [])
    if client_ip in disallowed:
        print(f"[ACCESS DENIED] {client_ip} заблокований у disallowed.")
        return False
    if allowed:
        if client_ip in allowed or client_ip in ("127.0.0.1", "::1", "localhost"):
            return True
        print(f"[ACCESS DENIED] {client_ip} не в списку allowed.")
        return False
    if client_ip in ("127.0.0.1", "::1", "localhost"):
        return True
    print(f"[ACCESS DENIED] Віддалений запит з {client_ip} відхилено.")
    return False
import os, sys, threading, time
from datetime import datetime
import socket
from fastapi import FastAPI, HTTPException, Request, UploadFile, File
from fastapi.responses import JSONResponse
import uvicorn
from pathlib import Path
from api.process_manager import ProcessManager
from common.project_manager import WorkDirs
def _set_my_work_dir():
    """[пояснення вилучено для скороченого лістингу]"""
    if getattr(sys, "frozen", False):
        default_dir = os.path.dirname(sys.executable)
    else:
        default_dir = os.path.abspath(os.path.join(os.path.dirname(__file__), "../"))
    if __name__ == "__main__" or os.environ.get("RUN_MAIN") == "true":
        print("=== 🧩 Запуск локальної копії SPM API Server ===")
        print(f"[INFO] Поточна тека: {default_dir}")
        if not sys.stdin.isatty():
            print("[INFO] Немає інтерактивного режиму — використовується поточна тека.")
            return default_dir
        print(
            "Введіть шлях до робочої теки (або натисніть Enter для використання поточної):"
        )
        user_dir = input("> ").strip()
        base_dir = user_dir or default_dir
        if not os.path.exists(base_dir):
            os.makedirs(base_dir)
            print(f"[INFO] Створено теку: {base_dir}")
        print(f"[INFO] Використовується шлях: {base_dir}")
        return base_dir
    return default_dir
base_dir = _set_my_work_dir()
work_dirs = WorkDirs(base_dir=base_dir)
manager = ProcessManager(work_dirs=work_dirs)
app = FastAPI(
    title="SPM API Server",
    description=("API для керування модулями SPM-системи."),
    version="0.2",
    docs_url="/__dev_docs",
    redoc_url="/__redoc_open_docs",
    openapi_url="/__openapi_docs.json",
)
@app.middleware(
    "http"
)
async def protect_docs(request: Request, call_next):
    if (
        request.url.path.startswith("/__dev_docs")
        or request.url.path.startswith("/__dev_docs")
        or request.url.path.startswith("/__openapi_docs.json")
    ):
        token = request.headers.get("X-Internal-Token")
        if token != "why_you_are_here_my_stupid_friend":
            raise HTTPException(status_code=403, detail="Access denied")
    return await call_next(request)
@app.post("/__get_my_work_dir", include_in_schema=False)
def _get_my_work_dir(request: Request):
    if not manager.check_ip_access(request):
        raise HTTPException(status_code=403, detail="Access denied (IP restricted)")
    prog_dir = manager._get_my_work_dir()
    return {prog_dir}
@app.post("/api/__testing_is_running", include_in_schema=False)
def test():
    """[пояснення вилучено для скороченого лістингу]"""
    return {"status": "ok", "message": "SPM API is running"}
@app.post("/api/__shutdown", include_in_schema=False)
def shutdown(request: Request):
    """[пояснення вилучено для скороченого лістингу]"""

raise RuntimeError('🛡 Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

    if not manager.check_ip_access(request):
        raise HTTPException(status_code=403, detail="Access denied (IP restricted)")
    print("[INFO] Отримано запит на завершення роботи сервера.")
    manager.shutdown()
    def delayed_exit():
        time.sleep(0.5)
        print("✅ Сервер успішно зупинено.")
        os._exit(0)
```

```
    threading.Thread(target=delayed_exit,
daemon=True).start()
    return {"status": "server_shutdown"}
@app.get("/api/ping")
async def ping():
    """[пояснення вилучено для скороченого
лістингу]"""
    return {
        "ok": True,
        "server": "SPM API Server",
        "version": "1.0.0",
        "status": "running",
        "ip":
socket.gethostbyname(socket.gethostname()),
        "port": 8765,
        "time": datetime.utcnow().isoformat() + "Z",
    }
@app.post("/api/upload")
async def upload_file(file: UploadFile = File(...)):
    cfg_ext                                          =
manager.extension_config.get("extensions", {})
    allowed_exts = set(cfg_ext.get("allowed", [".obj",
".json"]))
    max_size_mb = cfg_ext.get("max_size_mb", 10)
    ext = Path(file.filename).suffix.lower()
    if ext not in allowed_exts:
        raise HTTPException(
            status_code=400,
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
            detail=f"Недопустимий тип файлу '{ext}'.
Дозволено тільки: {', '.join(allowed_exts)}",
        )
    safe_name = os.path.basename(file.filename)
    dest = Path(work_dirs.upload_dir) / safe_name
    contents = await file.read()
    size_mb = len(contents) / (1024 * 1024)
    if size_mb > max_size_mb:
        raise                HTTPException(status_code=400,
detail=f"Файл перевищує {max_size_mb} МБ")
    with open(dest, "wb") as f:
        f.write(contents)
    relative_path                =                os.path.relpath(dest,
start=work_dirs.base_dir)
    return {"uploaded_to": relative_path, "size_mb":
round(size_mb, 2)}
@app.get("/api/modules")
def list_modules():
    """[пояснення вилучено для скороченого
лістингу]"""
    return manager.modules_config
@app.get("/api/status")
def get_status():
    """[пояснення вилучено для скороченого
лістингу]"""
    report = manager.get_status_report()
    return JSONResponse(content=report)
@app.post("/api/start/{module_name}")
async def start_module(module_name: str, request:
Request):
    """[пояснення вилучено для скороченого
лістингу]"""
    modules = manager.modules_config
    if module_name not in modules:
        raise HTTPException(
            status_code=404,                detail=f"Модуль
'{module_name}' не знайдено"
        )
    module = modules[module_name]
    if not module.get("enabled", True):
        raise                HTTPException(status_code=403,
detail=f"Модуль '{module_name}' вимкнено")
    try:
        payload = await request.json()
    except Exception:
        payload = {}
    manager.run_module(module_name, args=payload)
    return {
        "status": "started",
        "module": module_name,
        "args": payload,
        "entry": module.get("entry"),
    }
@app.post("/api/stop/{module_name}")
def stop_module(module_name: str):
    """[пояснення вилучено для скороченого
лістингу]"""
    manager.stop_process(module_name)
    return        {"status":        "stopped",        "module":
module_name}
@app.post("/api/restart/{module_name}")
def restart_module(module_name: str):
    """[пояснення вилучено для скороченого
лістингу]"""
    manager.stop_process(module_name)
    modules = manager.config.modules()
    if module_name in modules:
        entry = modules[module_name].get("entry")
        manager.start_process(module_name,  ["python3",
entry])
        return        {"status":        "restarted",        "module":
module_name}
    raise                HTTPException(status_code=404,
detail="Модуль не знайдено")
@app.post("/api/start_all", include_in_schema=False)
def start_all():
    """[пояснення вилучено для скороченого
лістингу]"""
    manager.run_all_enabled_modules()
    return {"status": "started_all"}
@app.post("/api/stop_all")
def stop_all():
    """[пояснення вилучено для скороченого
лістингу]"""
    manager.shutdown()

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

    return {"status": "all stopped"}
@app.get("/api/config")
def get_config():
    """[пояснення вилучено для скороченого
лістингу]"""
    return JSONResponse(content=manager.config.data)
if __name__ == "__main__":
    server_config = manager.config.server()
```

```
    host = server_config.get("host", "0.0.0.0")
    port = server_config.get("port", 8000)
    uvicorn.run("api.server:app", host=host, port=port, reload=False)
import sys, os, json
import tempfile
from pathlib import Path
def get_default_work_dirs(is_debug: bool = False):
    """[пояснення вилучено для скороченого лістингу]"""
    env_root = os.environ.get("PROJECT_ROOT")
    if env_root and os.path.exists(env_root):
        work_dirs = WorkDirs(base_dir=env_root, is_debug=is_debug)
        return work_dirs
    if getattr(sys, "frozen", False):
        base_dir = Path(sys.executable).parent
    else:
        base_dir = Path(__file__).resolve().parents[1]
    tempdir = Path(tempfile.gettempdir())
    if str(base_dir).startswith(str(tempdir)):
        if getattr(sys, "frozen", False):
            base_dir = Path(sys.executable).parent
        else:
            base_dir = Path.cwd()
    work_dirs = WorkDirs(base_dir=str(base_dir), is_debug=is_debug)
    return work_dirs
class WorkDirs:
    def __init__(self, base_dir: str, is_debug: bool = False):
        self.is_debug = is_debug
        self.base_dir = os.path.abspath(base_dir)
        if "MEI" in self.base_dir:
            print(f"[WARN] WorkDirs base_dir looks temporary: {self.base_dir}")
        self.json_dir = os.path.join(self.base_dir, "json")
        self.obj_dir = os.path.join(self.base_dir, "obj")
        self.upload_dir = os.path.join(self.base_dir, "upload")
        os.makedirs(self.json_dir, exist_ok=True)
        os.makedirs(self.obj_dir, exist_ok=True)
        os.makedirs(self.upload_dir, exist_ok=True)
        if self.is_debug:
            print(f"[DEBUG] WorkDirs created in {self.base_dir}")
    def __repr__(self):
        return f"WorkDirs(base={self.base_dir})"
class ConfigManager:
    """[пояснення вилучено для скороченого лістингу]"""
    def __init__(self, config_name: str, base_dir: str = "json"):
        self.data = {}
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.config_path = self.config_base_dir / config_name
        if self.config_path.exists():
            self.load()
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(self.config_path, "r", encoding="utf-8") as f:
            self.data = json.load(f)
    def save(self):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(self.config_path, "w", encoding="utf-8") as f:
            json.dump(self.data, f, ensure_ascii=False, indent=4)
    def get(self, *keys, default=None):
        """[пояснення вилучено для скороченого лістингу]"""
        node = self.data
        for key in keys:
            if isinstance(node, dict) and key in node:
                node = node[key]
            else:
                return default
        return node
    def set(self, *keys, value):
        """[пояснення вилучено для скороченого лістингу]"""
        node = self.data
        for key in keys[:-1]:
            node = node.setdefault(key, {})
        node[keys[-1]] = value
    def __repr__(self):
        return f"ConfigManager({self.config_path.name}, keys={len(self.data)})"
import argparse, sys, json
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
from pathlib import Path
from concurrent.futures import ProcessPoolExecutor, as_completed
from getangles.getangles import GetAngles
from objtoangles.objtoangles import ObjToAngles
from common.project_manager import ConfigManager, get_default_work_dirs
class DetermStatic:
    @staticmethod
    def _generate_new_path(output_file=None):
        """[пояснення вилучено для скороченого лістингу]"""
        if not output_file:
            output_file = "../obj/points.obj"
        x_min, x_max = -100, 100
        y_min, y_max = -100, 100
        z_min, z_max = 0, 100


raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---


        step = 5
        count = 0
        with open(output_file, "w", encoding="utf-8") as f:
            for z in range(z_min, z_max + 1, step):
                for y in range(y_min, y_max + 1, step):
                    for x in range(x_min, x_max + 1, step):
                        f.write(f"v {x} {z} {y}\n")
                        print(f"v {x} {z} {y}\n")
```

```
                count += 1
        print(f"[OK] Файл '{output_file}' створено ({count:,} точок)")
    @staticmethod
    def process_angle(point, angle, is_debug=False):
        """[пояснення вилучено для скороченого лістингу]"""
        alpha, beta, gamma = angle["alpha"], angle["beta"], angle["gamma"]
        x, y, z = point["x"], point["y"], point["z"]
        servo_angles = GetAngles(
            is_debug=is_debug,
            is_multi=True,
        ).calculation(alpha=alpha, beta=beta, gamma=gamma)
        valid = not all(np.isnan(value) for value in servo_angles)
        result_entry = {
            "point": {"x": x, "y": y, "z": z},
            "angle": {"alpha": alpha, "beta": beta, "gamma": gamma},
            "servo": {
                "s1": servo_angles[0],
                "s2": servo_angles[1],
                "s3": servo_angles[2],
            },
            "valid": valid,
        }
        return result_entry
    @staticmethod
    def filter_and_visualize_points(
        input_json="filtered.json", output_json="valid_result.json"
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(input_json, "r", encoding="utf-8") as f:
            data = json.load(f)
        points = np.array(
            [[p["x"], p["y"], p["z"]] for p in data["points"]], dtype=float
        )
        angles = np.array(
            [[a["alpha"], a["beta"], a["gamma"]] for a in data["angles"]], dtype=float
        )
        if len(points) < 4:
            print("Недостатньо точок для побудови опуклої оболонки.")
            return
        hull = ConvexHull(points)
        surface_indices = np.unique(hull.vertices)
        valid_points = points[surface_indices]
        valid_angles = angles[surface_indices]
        result = {
            "points": [
                {"x": float(x), "y": float(y), "z": float(z)}
                for x, y, z in valid_points
            ],
            "angles": [
                {"alpha": float(alpha), "beta": float(beta), "gamma": float(gamma)}
                for alpha, beta, gamma in valid_angles
            ],
        }
        with open(output_json, "w", encoding="utf-8") as f_out:
            json.dump(result, f_out, ensure_ascii=False, indent=2)
        print(f"[OK] Збережено {len(valid_points)} точок у {output_json}")
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        xs, ys, zs = valid_points[:, 0], valid_points[:, 1], valid_points[:, 2]
        ax.scatter(xs, ys, zs, color="red", s=50, label="Valid points")
        # З'єднуємо всі точки між собою лініями
        for i in range(len(valid_points)):
            for j in range(i + 1, len(valid_points)):
                ax.plot(
                    [valid_points[i, 0], valid_points[j, 0]],
                    [valid_points[i, 1], valid_points[j, 1]],
                    [valid_points[i, 2], valid_points[j, 2]],
                    color="blue",
                    linewidth=0.5,
                    alpha=0.4,
                )
        ax.set_xlabel("X")
        ax.set_ylabel("Y")
        ax.set_zlabel("Z")
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
        ax.legend()
        plt.title("Робоча область — зовнішні точки, з'єднані між собою")
        plt.show()
class AreaConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.1
    def __init__(self, config_name="area_config.json", work_dirs=None, is_debug=False):
        self.work_dirs = work_dirs
        self.is_debug = is_debug
        base_dir = work_dirs.json_dir if work_dirs else "json"
        super().__init__(config_name, base_dir)
        if not self.config_path.exists():
            self.ensure_default_config()
    def ensure_default_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_data = {

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            "version": self.CURRENT_VERSION,
            "last_used": {
                "input_obj": "../obj/points.obj",
                "conv_out": "../json/conv_output.json",
                "valid_out": "../json/valid.json",
                "valid_res": "../json/valid_result.json",
                "output": "../json/output.json",
                "unoutput": "../json/unoutput.json",
            },
            "stats": {},
```

```
        }
        self.config_base_dir.mkdir(parents=True,
exist_ok=True)
        with open(self.config_path, "w", encoding="utf-
8") as f:
            json.dump(default_data, f, ensure_ascii=False,
indent=4)
        self.data = default_data
        if self.is_debug:
            print(f"[DEBUG] Created default area config:
{self.config_path}")
    @staticmethod
    def load_data(path):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        if not path.exists():
            raise FileNotFoundError(f"JSON file not found:
{path}")
        with open(path, "r", encoding="utf-8") as f:
            return json.load(f)
    @staticmethod
    def save_json(path, data):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False,
indent=2)
        print(f"[OK] Saved JSON → {path}")
    @staticmethod
    def save_obj(path, vertices):
        """[пояснення вилучено для скороченого
лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, "w", encoding="utf-8") as f:
            f.write("\n".join(vertices))
        print(f"[OK] Saved OBJ ({len(vertices)} points) →
{path}")
class DetermWorkArea:
    def __init__(
        self,
        input_obj: str = "points.obj",
        conv_out: str = "conv_output.json",
        valid_out: str = "valid.json",
        valid_res: str = "valid_result.json",
        output: str = "output.json",
        unoutput: str = "unoutput.json",
        *,
        gen_new_path: bool = False,
        is_new_path: bool = False,
        is_validate: bool = True,
        is_simpify: bool = False,
        is_check: bool = True,
        is_debug: bool = False,
    ):
        super().__init__()
        self.gen_new_path = gen_new_path
        self.is_new_path = is_new_path
        self.is_validate = is_validate
        self.is_simpify = is_simpify
        self.is_check = is_check
        self.is_debug = is_debug
        self.work_dirs = get_default_work_dirs()
        self.json_dir = Path(self.work_dirs.json_dir)
        self.obj_dir = Path(self.work_dirs.obj_dir)
        self.input_obj_path = str(self.obj_dir / input_obj)
        self.conv_out_path = str(self.json_dir / conv_out)
        self.valid_out_path = str(self.json_dir / valid_out)
        self.valid_res = str(self.json_dir / valid_res)
        self.output_path = str(self.json_dir / output)
        self.unoutput_path = str(self.json_dir / unoutput)
        self.ac = AreaConfig(work_dirs=self.work_dirs,
is_debug=self.is_debug)
        if self.gen_new_path:

DetermStatic._generate_new_path(self.input_obj_path)
        if self.is_new_path:
            self._convert_obj_to_angles()
        if self.is_validate:
            self._validate_angles()
        if self.is_simpify:
            DetermStatic.filter_and_visualize_points(
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
                self.valid_out_path, self.valid_res
            )
        if self.is_check:
            self.check_perm_angles()
    def _convert_obj_to_angles(self):
        converter = ObjToAngles(
            self.input_obj_path, self.conv_out_path,
is_filter=False
        )
        converter.run()
    def _validate_angles(self):
        data = self.ac.load_data(self.conv_out_path)
        points, angles = data["points"], data["angles"]
        result = {"points": [], "angles": []}
        for point, angle in zip(points, angles):
            if (

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

                any(abs(angle[k]) > 46 for k in ["alpha",
"beta"])
                or abs(angle["gamma"]) > 180
            ):
                continue
            result["points"].append(point)
            result["angles"].append(angle)
            if self.is_debug:
                print(f"{point} → {angle}")
        self.ac.save_json(self.valid_out_path, result)
    def check_perm_angles(self):
        result = {"points": [], "angles": [], "servo": []}
        unresult = {"points": [], "angles": [], "servo": []}
        obj_out_file = []
        obj_unout_file = []
        path = self.valid_res if self.is_simpify else
self.valid_out_path
        data = self.ac.load_data(path)
        points, angles = data["points"], data["angles"]
        with ProcessPoolExecutor() as executor:
            futures = [
```

```
            executor.submit(DetermStatic.process_angle,
point, angle, self.is_debug)
            for point, angle in zip(points, angles)
        ]
        for future in as_completed(futures):
            entry = future.result()
            x, y, z = entry["point"].values()
            if entry["valid"]:
                result["points"].append(entry["point"])
                result["angles"].append(entry["angle"])
                result["servo"].append(entry["servo"])
                obj_out_file.append(f"v {x:.3f} {z:.3f}
{y:.3f}")
            else:
                unresult["points"].append(entry["point"])
                unresult["angles"].append(entry["angle"])
                unresult["servo"].append(entry["servo"])
                obj_unout_file.append(f"v {x:.3f} {z:.3f}
{y:.3f}")
    self.ac.save_json(self.output_path, result)
    self.ac.save_json(self.unoutput_path, unresult)
    self.ac.save_obj(self.obj_dir /
"obj_work_area.obj", obj_out_file)
    self.ac.save_obj(self.obj_dir /
"obj_unwork_area.obj", obj_unout_file)
  @staticmethod
  def build_parser():
    epilog_text = """[пояснення вилучено для
скороченого лістингу]"""
    parser = argparse.ArgumentParser(
        prog="determ_work_area",
        description=(
            "Комплексний розрахунок робочої зони
SPM: "
            "генерація, конвертація OBJ→JSON,
валідація та обчислення кутів сервоприводів."
        ),

formatter_class=argparse.RawDescriptionHelpFormatt
er,
        epilog=epilog_text,
    )
    parser.add_argument(
        "--input_obj",
        type=str,
        default="points.obj",
        help="Вхідний OBJ файл (у директорії obj/).",
    )
    parser.add_argument(
        "--conv_out",
        type=str,
        default="conv_output.json",
        help="Вихідний JSON після конвертації
OBJ→ANGLES.",
    )
    parser.add_argument(
        "--valid_out",
        type=str,
        default="valid.json",
        help="Файл відфільтрованих точок після
перевірки кутів α, β, γ.",
    )
    parser.add_argument(
        "--valid_res",
        type=str,
        default="valid_result.json",
        help="Файл результатів після спрощення
(simplify).",
    )
    parser.add_argument(
        "--output",
        type=str,
        default="output.json",
        help="Файл робочих точок (успішні
розрахунки).",
    )
    parser.add_argument(
        "--unoutput",
        type=str,
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
        default="unoutput.json",
        help="Файл неробочих точок (невдалі
розрахунки).",
    )
    parser.add_argument(
        "--gen_new_path",
        action=argparse.BooleanOptionalAction,
        default=False,
        help="Генерувати новий OBJ шлях для
тесту.",
    )
    parser.add_argument(
        "--is_new_path",
        action=argparse.BooleanOptionalAction,
        default=False,
        help="Конвертувати OBJ→ANGLES через
ObjToAngles.",

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

    )
    parser.add_argument(
        "--is_validate",
        action=argparse.BooleanOptionalAction,
        default=True,
        help="Фільтрувати кути αβγ після
конвертації.",
    )
    parser.add_argument(
        "--is_simpify",
        action=argparse.BooleanOptionalAction,
        default=False,
        help="Спрощувати результати після валідації
(викликає filter_and_visualize_points).",
    )
    parser.add_argument(
        "--is_check",
        action=argparse.BooleanOptionalAction,
        default=True,
        help="Виконати розрахунок кутів
сервоприводів (через GetAngles).",
    )
    parser.add_argument(
        "--is_debug",
        action=argparse.BooleanOptionalAction,
        default=False,
```

```
        help="[DEBUG_INFO] Виводити проміжні повідомлення.",
      )
      return parser
   @staticmethod
   def run_terminal():
      parser = DetermWorkArea.build_parser()
      if len(sys.argv) == 1:
         parser.print_help()
         sys.exit(0)
      args = parser.parse_args()
      app = DetermWorkArea(
         input_obj=args.input_obj,
         conv_out=args.conv_out,
         valid_out=args.valid_out,
         valid_res=args.valid_res,
         output=args.output,
         unoutput=args.unoutput,
         gen_new_path=args.gen_new_path,
         is_new_path=args.is_new_path,
         is_validate=args.is_validate,
         is_simpify=args.is_simpify,
         is_check=args.is_check,
         is_debug=args.is_debug,
      )
      if args.is_debug:
         print("[OK] DetermWorkArea finished successfully.")
      return app
if __name__ == "__main__":
   if len(sys.argv) > 1:
      DetermWorkArea.run_terminal()
   else:
      app = DetermWorkArea(
         gen_new_path=True,
         is_new_path=True,
         is_validate=True,
         is_simpify=False,
         is_check=True,
         is_debug=False,
      )
"""[пояснення вилучено для скороченого лістингу]"""
import os, sys, re, json, shutil
import argparse
import numpy as np
from numpy import arcsin
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
import itertools
from mpl_toolkits.mplot3d import Axes3D
from pathlib import Path
from common.project_manager import ConfigManager, get_default_work_dirs
"""[пояснення вилучено для скороченого лістингу]"""
class MathFunction:
   @staticmethod
   def sin_alpha(a, b):
      """[пояснення вилучено для скороченого лістингу]"""
      return (a / b**2) * np.sqrt(b**2 - (a**2) / 4)
   @staticmethod
   def get_vector(bx0, by0, bx, by):
      """[пояснення вилучено для скороченого лістингу]"""
      if bx0 == bx and by0 == bx:
         return 0
      return np.sqrt((bx - bx0) ** 2 + (by - by0) ** 2)
class AnglesConfig(ConfigManager):
   """[пояснення вилучено для скороченого лістингу]"""
   CURRENT_VERSION = 0.1
   def __init__(
      self,
      config_name="angles_config.json",
      work_dirs=None,
      template_name="angles_config.json.default",
      is_multi: bool = False,
   ):
      self.work_dirs = work_dirs
      self.template_name = template_name
      self.prefix = "angles_config"
      self.max_attempts = 10000


raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---


      self.is_multi = is_multi
      base_dir = work_dirs.json_dir if work_dirs is not None else "json"
      self.base_dir = base_dir
      self.config_path = Path(base_dir) / config_name
      self.config_base_dir = Path(base_dir)
      self.config_base_dir.mkdir(parents=True, exist_ok=True)
      self.template_path = self.config_base_dir / self.template_name
      if not self.template_path.exists():
         self._save_default_config_template()
      if self.is_multi:
         self.create_instance_config()
      else:
         self.ensure_config()
      super().__init__(
         config_name=self.config_path.name, base_dir=str(self.config_base_dir)
      )
      self.load()
   def _save_default_config_template(self):
      """[пояснення вилучено для скороченого лістингу]"""
      default_config = self._get_default_structure()
      with open(self.template_path, "w", encoding="utf-8") as f:
         json.dump(default_config, f, ensure_ascii=False, indent=4)
      if self.work_dirs and self.work_dirs.is_debug:
         print(f"[DEBUG] Created default config: {self.config_path}")
   def ensure_config(self):
      """[пояснення вилучено для скороченого лістингу]"""
      if self.config_path.exists():
         return
      if not self.template_path.exists():
         self._save_default_config_template()
```

```
        try:
            shutil.copyfile(self.template_path,
self.config_path)
            with open(self.config_path, "r", encoding="utf-
8") as f:
                self.data = json.load(f)
        except (json.JSONDecodeError,
FileNotFoundError, OSError) as e:
            if self.work_dirs and self.work_dirs.is_debug:
                print(
                    f"[WARNING] template copy/load failed
({e}), writing default structure"
                )
            default_config = self._get_default_structure()
            with open(self.config_path, "w", encoding="utf-
8") as f:
                json.dump(default_config, f,
ensure_ascii=False, indent=4)
            self.data = default_config
            if self.work_dirs and self.work_dirs.is_debug:
                print(f"[DEBUG] Created config:
{self.config_path}")
    def validate_structure(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        required_keys = {
            "version",
            "coordinates",
            "angles",
        }
        if not isinstance(self.data, dict):
            return False
        if not required_keys.issubset(self.data.keys()):
            return False
        version = self.data.get("version")
        if not isinstance(version, (float, int)):
            return False
        try:
            _ = self.data["angles"]["platform"]["zero"]
            _ = self.data["coordinates"]["platform"]["zero"]
        except (KeyError, TypeError):
            return False
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
        return True
    def load(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        super().load()
        if not self.validate_structure():
            print(
                f"[WARNING] Invalid config structure in
{self.config_path}, resetting..."
            )
            self.ensure_config()
        elif self.data["version"] <
self.CURRENT_VERSION:
            print(
                f"[INFO] Updating config version from
{self.data['version']} to {self.CURRENT_VERSION}"
            )
            self.data["version"] =
self.CURRENT_VERSION
            self.save()
    def create_instance_config(self):
        if not self.template_path.exists():
            self._save_default_config_template()
        with open(self.template_path, "rb") as f:
            template_bytes = f.read()
        pattern =
re.compile(rf"^{re.escape(self.prefix)}\.(\d+)\.json$")
        existing = [
            int(m.group(1))
            for fname in os.listdir(self.config_base_dir)
            if (m := pattern.match(fname))
        ]
        next_number = max(existing) + 1 if existing else 1
        for _ in range(self.max_attempts):
            dest_name =
f"{self.prefix}.{next_number}.json"
            dest_path = self.config_base_dir / dest_name
            try:
                with open(dest_path, "xb") as out:
                    out.write(template_bytes)
                self.config_path = dest_path


raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---


                with open(dest_path, "r", encoding="utf-8")
as f:
                    self.data = json.load(f)
                if self.work_dirs and self.work_dirs.is_debug:
                    print(f"[DEBUG] Created new config:
{dest_path}")
                return dest_path
            except FileExistsError:
                next_number += 1
        raise RuntimeError(
            f"Unable to create unique config file after
{self.max_attempts} attempts"
        )
    def update_value(self, path_str, new_value):
        keys = path_str.split(".")
        d = self.data
        for k in keys[:-1]:
            d = d[k]
        d[keys[-1]] = new_value
        self.save()
        if self.work_dirs and self.work_dirs.is_debug:
            print(f"[DEBUG] Updated {path_str} =
{new_value}")
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        return {
            "version": self.CURRENT_VERSION,
            "coordinates": {
                "platform": {
                    "zero": {
                        "left": {
                            "c": [40.0, 0.0, 55.0],
                            "c2": [-20, -34.64, 55.0],
                            "c3": [-20, 34.64, 55.0],
                        },
                        "right": {
                            "c": [40.0, 0.0, 55.0],
```

```
                    "c2": [-20, -34.64, 55.0],
                    "c3": [-20, 34.64, 55.0],
                },
            },
            "last": {},
            "all": [],
        },
        "middle": {
            "zero": {
                "left": {
                    "b": [18.75, -46.35124054],
                    "b2": [-49.5161937, 6.93877234],
                    "b3": [30.7662487, 39.41367708],
                },
                "right": {
                    "b": [18.75, 46.35124054],
                    "b2": [30.7662487, -39.41367708],
                    "b3": [-49.5161937, -6.93877234],
                },
            },
            "last": {},
            "all": [],
        },
    },
    "angles": {
        "platform": {
            "zero": {"alpha": 0, "beta": 0, "gamma": 0},
            "last": {},
            "all": [],
        },
        "servo": {
            "zero": {"s1": 90, "s2": 90, "s3": 90},
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
            "last": {},
            "all": [],
        },
    },
}
class GetAngles:
    def __init__(
        self,
        *,
        is_bend: bool = True,
        is_debug: bool = False,
        is_multi: bool = False,
        rc: float = 75.0,
        r1: float = 50.0,
        is_2d: bool = False,
        is_3d: bool = False,
    ):
        self.work_dirs = get_default_work_dirs()
        self.Rc = rc
        self.R1 = r1
        self.is_bend = is_bend
        self.is_debug = is_debug
        self.is_multi = is_multi
        self.is_2d = is_2d
        self.is_3d = is_3d
        self.alpha = self.beta = self.gamma = 0.0
        self.is_p_last_coords = False
        self.is_m_last_coords = False
        self.is_p_last_angles = False
        self.is_s_last_angles = False
        self.flag_map = {
            "coords": {
                "platform": "is_p_last_coords",

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

                "middle": "is_m_last_coords",
            },
            "angles": {
                "platform": "is_p_last_angles",
                "servo": "is_s_last_angles",
            },
        }
        self.get_config()
        self.p_coords, self.m_coords, self.candidates = [], [], []
    def calculation(self, alpha=None, beta=None, gamma=None):
        if alpha is not None:
            self.alpha = alpha
        if beta is not None:
            self.beta = beta
        if gamma is not None:
            self.gamma = gamma
        if not self.is_multi:
            self.p_coords.clear()
            self.m_coords.clear()
            self.candidates.clear()
        self.p_last_coords = self.get_p_last_coords()
        self.m_last_coords = self.get_m_last_coords()
        self.p_last_angles = self.get_p_last_angles()
        self.s_last_angles = self.get_s_last_angles()
        for i, p_last_coord in enumerate(self.p_last_coords):
            p_coord = self.get_p_coords(coords=p_last_coord)
            m_coord = self.get_m_cords(p_coord)
            if m_coord is None:
                if self.is_debug:
                    print(
                        f"⚠️ Пропущено розрахунок для поточного
                        f"(α,β,γ) = ({self.alpha}, {self.beta}, {self.gamma}) — "
                        "отримано уявні або вироджені координати при побудові middle."
                    )
                return (np.nan, np.nan, np.nan)
            bx1, bx2, by1, by2 = m_coord
            self.p_coords.append(p_coord)
            self.m_coords.append(m_coord)
            self.candidates.append([(bx1, by1), (bx1, by2), (bx2, by1), (bx2, by2)])
        if self.is_debug:
            print(self.candidates[0])
            print(self.candidates[1])
            print(self.candidates[2])
        valid_combos = [
            self.filter_points_on_circle(cand, self.R1) for cand in self.candidates
        ]
        if not all(valid_combos):
```

```
            if self.is_debug:
                sizes = [len(v) for v in valid_combos]
                print(
                    "⚠️ Після перевірки належності до кола одна з груп кандидатів пуста — "
                    f"valid sizes = {sizes}. Пропускаємо набір alpha/beta/gamma."
                )
            return (np.nan, np.nan, np.nan)
        valid_combos = self.filter_valid_combinations(tuple(valid_combos), min_dist=10)
        if not valid_combos:
            if self.is_debug:
                print("⚠️ Після фільтрації за відстанню — valid_combos порожній.")
            return (np.nan, np.nan, np.nan)
        if self.is_2d:
            self.plot_combinations(
                valid_combos,
                self.p_coords,
                decimals=2,
            )
        valid_combos = [
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
            combo
            for combo in valid_combos
            if self.is_valid_combo(
                combo[0],
                combo[1],
                combo[2],
                self.p_coords,
            )
        ]
        if not valid_combos:
            if self.is_debug:
                print("⚠️ Після перевірки перетинів — valid_combos порожній.")
            return (np.nan, np.nan, np.nan)
        if self.is_2d:
            self.plot_combinations(
                valid_combos,
                self.p_coords,
                decimals=2,
            )
        valid_combos = self.filter_by_bend_side(valid_combos, self.p_coords)
        closest_combo = self.choose_closest_combo(
            valid_combos,
            self.m_last_coords,
        )
        valid_combos = [closest_combo] if closest_combo else []
        if not valid_combos:
            if self.is_debug:
                print("⚠️ Після перевірки перетинів — valid_combos порожній.")
            return (np.nan, np.nan, np.nan)
        if self.is_2d:
            self.plot_combinations(
                valid_combos,
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

                self.p_coords,
                decimals=2,
            )
        a, a2, a3 = self.get_angles(
            self.m_last_coords,
            valid_combos,
        )
        dir_b = self.circle_direction(
            m_last_coords=self.m_last_coords[0], valid_combos=valid_combos, k=0
        )
        dir_b2 = self.circle_direction(
            m_last_coords=self.m_last_coords[1], valid_combos=valid_combos, k=1
        )
        dir_b3 = self.circle_direction(
            m_last_coords=self.m_last_coords[2], valid_combos=valid_combos, k=2
        )
        self.m_coords_valid = np.array(valid_combos[0])
        if self.is_3d:
            self.plot_mechanism_3d(
                self.p_last_coords,
                self.m_last_coords,
                self.p_coords,
                a,
                a2,
                a3,
                valid_combos=valid_combos,
                axis_range=70,
            )
        self.s_angles_1, self.s_angles_2, self.s_angles_3 = self.s_angles(
            self.s_last_angles[0],
            self.s_last_angles[1],
            self.s_last_angles[2],
            dir_b,
            dir_b2,
            dir_b3,
            a,
            a2,
            a3,
        )
        status_msg = (
            f"✅ Обрана комбінація: {closest_combo}"
            if closest_combo
            else "⚠️ Немає валідних комбінацій після фільтра"
        )
        if self.is_debug:
            print(
                f"Останні координати платформи:\nC: {self.p_last_coords[0]},\nC2: {self.p_last_coords[1]},\nC3: {self.p_last_coords[2]}\n"
                f"Останні координати middle:\nB: {self.m_last_coords[0]},\nB2: {self.m_last_coords[1]},\nB3: {self.m_last_coords[2]}\n"
```

```
            f"Нові координати платформи: {self.p_coords[0]}, {self.p_coords[1]}, {self.p_coords[2]}\n"
            f"{valid_combos}\n"
            f"{status_msg}\n"
            f"{a}, {a2}, {a3}\n"
            f"Знак:\nДля A: {dir_b},\nДля A2: {dir_b2},\nДля A3: {dir_b3}"
            f"Останні кути платформи:\nC: {self.p_last_angles[0]},\nC2: {self.p_last_angles[1]},\nC3: {self.p_last_angles[2]}\n"
            f"Останні кути servo:\nB: {self.s_last_angles[0]},\nB2: {self.s_last_angles[1]},\nB3: {self.s_last_angles[2]}\n"
        )
        print(f"{self.s_angles_1}, {self.s_angles_2}, {self.s_angles_3}")
        self.update_last_coords(
            "platform", self.p_coords[0], self.p_coords[1], self.p_coords[2]
        )
        self.update_last_coords(
            "middle",
            self.m_coords_valid[0],
            self.m_coords_valid[1],
            self.m_coords_valid[2],
        )
        self.update_last_angles("platform", self.alpha, self.beta, self.gamma)
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
        self.update_last_angles(
            "servo", self.s_angles_1, self.s_angles_2, self.s_angles_3
        )
        return (self.s_angles_1, self.s_angles_2, self.s_angles_3)
    def p_absolute_angles(
        self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode="delta"
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        if mode == "delta":
            p_absolute_angles_1 = p_last_angles_1 + self.alpha
            p_absolute_angles_2 = p_last_angles_2 + self.beta
            p_absolute_angles_3 = p_last_angles_3 + self.gamma
        elif mode == "absolute":
            p_absolute_angles_1 = self.alpha
            p_absolute_angles_2 = self.beta
            p_absolute_angles_3 = self.gamma
        else:
            raise ValueError("mode має бути 'delta' або 'absolute'")
        p_absolute_angles_1 = (p_absolute_angles_1 + 360) % 360
        p_absolute_angles_2 = (p_absolute_angles_2 + 360) % 360
        p_absolute_angles_3 = (p_absolute_angles_3 + 360) % 360
        return p_absolute_angles_1, p_absolute_angles_2, p_absolute_angles_3
    def p_angles(
        self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode="delta"
    ):
        if mode == "delta":
            p_angles_1 = self.alpha
            p_angles_2 = self.beta
            p_angles_3 = self.gamma
        elif mode == "absolute":
            p_angles_1 = self.alpha - p_last_angles_1

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            p_angles_2 = self.beta - p_last_angles_2
            p_angles_3 = self.gamma - p_last_angles_3
        else:
            raise ValueError("mode має бути 'delta' або 'absolute'")
        return p_angles_1, p_angles_2, p_angles_3
    @staticmethod
    def s_angles(
        s_last_angles_1,
        s_last_angles_2,
        s_last_angles_3,
        dir_b,
        dir_b2,
        dir_b3,
        a,
        a2,
        a3,
    ):
        s_angles_1 = s_last_angles_1 + dir_b * a
        s_angles_2 = s_last_angles_2 + dir_b2 * a2
        s_angles_3 = s_last_angles_3 + dir_b3 * a3
        return s_angles_1, s_angles_2, s_angles_3
    def plot_mechanism_3d(
        self,
        p_last_coords,
        m_last_coords,
        p_coords,
        a: float,
        a2: float,
        a3: float,
        *,
        valid_combos: list | None = None,
        title: str = "3D Visualization",
        axis_range: float = 120,
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        p_last_coords_1, p_last_coords_2, p_last_coords_3 = (
            p_last_coords[0],
            p_last_coords[1],
            p_last_coords[2],
        )
        m_last_coords_1, m_last_coords_2, m_last_coords_3 = (
            m_last_coords[0],
            m_last_coords[1],
```

```
        m_last_coords[2],
    )
    p_coords_1, p_coords_2, p_coords_3 = p_coords[0], p_coords[1], p_coords[2]
    m_coords_1, m_coords_2, m_coords_3 = valid_combos[0]
    fig = plt.figure(figsize=(14, 10))
    ax = fig.add_subplot(111, projection="3d")
    style = {
        "platform_old": {"color": "red", "lw": 2},
        "platform_new": {"color": "darkred", "lw": 2, "ls": "--"},
        "link_new": {"color": "green", "lw": 2},
        "link_old": {"color": "blue", "lw": 1, "ls": "--"},
        "link_aux": {"color": "gray", "lw": 1, "ls": "--"},
        "circle_P": {"color": "purple", "ls": "--", "lw": 1.5},
        "circle_O": {"color": "orange", "ls": ":", "lw": 1.5},
        "points": {
            "O": "black",
            "P": "purple",
            "C": "red",
            "C_new": "darkred",
            "B": "blue",
            "B_new": "green",
        },
    }
    def to3d(p: np.ndarray, z: float = 0) -> np.ndarray:
        return np.array([p[0], p[1], z]) if len(p) == 2 else np.array(p)
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
    def draw_line(p1: np.ndarray, p2: np.ndarray, style_key="platform") -> None:
        s = style[style_key]
        ax.plot([p1[0], p2[0]], [p1[1], p2[1]], [p1[2], p2[2]], **s)
    def draw_point(p: np.ndarray, label: str, style_key="O") -> None:
        ax.scatter(
            p[0], p[1], p[2], s=50, c=style["points"][style_key], label=label
        )
    def draw_circle(
        center: np.ndarray,
        r: float,
        z_fixed: float | None = None,
        style_key="circle_P",
    ) -> None:
        theta = np.linspace(0, 2 * np.pi, 200)
        x = center[0] + r * np.cos(theta)
        y = center[1] + r * np.sin(theta)
        z = np.full_like(theta, z_fixed if z_fixed is not None else center[2])
        ax.plot(x, y, z, **style[style_key])
    C, C2, C3 = p_last_coords_1, p_last_coords_2, p_last_coords_3
    Cn, C2n, C3n = p_coords_1, p_coords_2, p_coords_3
    B, B2, B3 = to3d(m_last_coords_1), to3d(m_last_coords_2), to3d(m_last_coords_3)
    Bn, B2n, B3n = to3d(m_coords_1), to3d(m_coords_2), to3d(m_coords_3)
    O = np.array([0, 0, 0])
    P = np.array([0, 0, 55])
    for b, c in [(B, C), (B2, C2), (B3, C3)]:
        draw_line(O, b, "link_old")
        draw_line(b, c, "platform_old")
    for b_new, c_new in [(Bn, Cn), (B2n, C2n), (B3n, C3n)]:
        draw_line(O, b_new, "link_aux")
        draw_line(b_new, c_new, "link_new")

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

    for p1, p2 in [(C, C2), (C2, C3), (C3, C)]:
        draw_line(p1, p2, "platform_old")
    for p1, p2 in [(Cn, C2n), (C2n, C3n), (C3n, Cn)]:
        draw_line(p1, p2, "platform_new")
    for b in [B, B2, B3]:
        draw_line(O, b, "link_old")
    draw_point(O, "O", "O")
    draw_point(P, "P", "P")
    draw_point(C, "C", "C")
    draw_point(C2, "C2", "C")
    draw_point(C3, "C3", "C")
    draw_point(Cn, "C new", "C_new")
    draw_point(C2n, "C2 new", "C_new")
    draw_point(C3n, "C3 new", "C_new")
    draw_point(B, "B", "B")
    draw_point(B2, "B2", "B")
    draw_point(B3, "B3", "B")
    draw_point(Bn, "B new", "B_new")
    draw_point(B2n, "B2 new", "B_new")
    draw_point(B3n, "B3 new", "B_new")
    draw_circle(O, self.R1, z_fixed=O[2], style_key="circle_O")
    draw_circle(P, 40, z_fixed=P[2], style_key="circle_P")
    axis_len = axis_range * 1.1
    ax.quiver(0, 0, 0, axis_len, 0, 0, color="r", arrow_length_ratio=0.1)
    ax.quiver(0, 0, 0, 0, axis_len, 0, color="g", arrow_length_ratio=0.1)
    ax.quiver(0, 0, 0, 0, 0, axis_len, color="b", arrow_length_ratio=0.1)
    ax.text(axis_len, 0, 0, "X", color="r")
    ax.text(0, axis_len, 0, "Y", color="g")
    ax.text(0, 0, axis_len, "Z", color="b")
    text_angles = (
        f"alpha={self.alpha:.1f}°,\n"
        f"beta={self.beta:.1f}°,\n"
        f"gamma={self.gamma:.1f}°.\n\n"
        f"a={a:.1f}°,\n"
        f"a2={a2:.1f}°,\n"
        f"a3={a3:.1f}°."
    )
    ax.text2D(0.02, 0.95, text_angles, transform=ax.transAxes, fontsize=10)
    ax.set_title(title)
    ax.set_xlabel("X")
    ax.set_ylabel("Y")
    ax.set_zlabel("Z")
    ax.set_box_aspect([1, 1, 1])
    ax.set_xlim(-axis_range, axis_range)
```

```
        ax.set_ylim(-axis_range, axis_range)
        ax.set_zlim(0, axis_range)
        plt.legend(loc="center left",
bbox_to_anchor=(1.05, 0.5))
        plt.tight_layout()
        plt.show()
    def _get_last_angles(self, angles_type: str) ->
tuple[float, float, float]:
        """[пояснення вилучено для скороченого
лістингу]"""
        mapping = {
            "platform": ["alpha", "beta", "gamma"],
            "servo": ["s1", "s2", "s3"],
        }
        if angles_type not in mapping:
            raise ValueError(
                f"Unknown group '{angles_type}'.
Очікується 'platform' або 'servo'."
            )
        keys = mapping[angles_type]
        is_last_flag = getattr(
            self, f"{self.flag_map['angles'][angles_type]}"
        )
        angles_key = "zero" if not is_last_flag else "last"
        values
[self.config.data["angles"][angles_type][angles_key][k]
for k in keys]
        if self.is_debug:
            print(f"[DEBUG] values ({angles_type} angles,
{angles_key}): {values}")
        if not is_last_flag:
            self.update_last_angles(angles_type, *values)
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
            self.config.load()
            setattr(self,
f"{self.flag_map['angles'][angles_type]}", True)
        return tuple(values)
    def get_p_last_angles(self) -> tuple[float, float,
float]:
        """[пояснення вилучено для скороченого
лістингу]"""
        return self._get_last_angles("platform")
    def get_s_last_angles(self) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого
лістингу]"""
        return self._get_last_angles("servo")
    def circle_direction(self, m_last_coords,
valid_combos, k):
        """[пояснення вилучено для скороченого
лістингу]"""
        det_x, det_y = self.determ_num(
            m_last_coords=m_last_coords,
valid_combos=valid_combos, k=k
        )
        if self.is_debug:
            debug_message = (
                f"Знаки A{" if k == 0 else f"_{k + 1}"}: 'x' -
{det_x}, 'y' - {det_y}"
            )
            print(debug_message)
        if det_x == 0 and det_y == 0:
            return 0
        x, y = valid_combos[0][k]
        if x >= 0 and y >= 0:
            return +1 if det_x < 0 or det_y > 0 else -1
        elif x < 0 and y >= 0:
            return +1 if det_x < 0 or det_y < 0 else -1
        elif x < 0 and y < 0:
            return +1 if det_x > 0 or det_y < 0 else -1
        else:

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

            return +1 if det_x > 0 or det_y > 0 else -1
    @staticmethod
    def choose_closest_combo(combos, last_coords):
        """[пояснення вилучено для скороченого
лістингу]"""
        if not combos:
            return None
        def dist(a, b):
            return np.linalg.norm(np.array(a) - np.array(b))
        best_combo = min(
            combos,
            key=lambda combo: sum(dist(c, l) for c, l in
zip(combo, last_coords)),
        )
        return best_combo
    def get_p_last_coords(self):
        return self._get_last_coords(kind="platform",
recalc=False)
    def get_m_last_coords(self):
        return self._get_last_coords(kind="middle",
recalc=False)
    def _get_last_coords(self, *, kind: str, recalc: bool =
False):
        """[пояснення вилучено для скороченого
лістингу]"""
        mapping = {
            "platform": ["c", "c2", "c3"],
            "middle": ["b", "b2", "b3"],
        }
        if kind not in mapping:
            raise ValueError(
                f"Unknown kind '{kind}'. Очікується
'platform' або 'middle'."
            )
        keys = mapping[kind]
        is_last_flag = getattr(
            self, f"{self.flag_map['coords'][kind]}"
        )
        coords_key = "zero" if not is_last_flag else "last"
        if coords_key == "zero":
            side = "right" if self.is_bend else "left"
            values = [

self.config.data["coordinates"][kind][coords_key][side]
[k] for k in keys
            ]
        else:
            values = [

self.config.data["coordinates"][kind][coords_key][k]
for k in keys
            ]
```

```
        if self.is_debug:
            print(f"[DEBUG] values ({kind}): {values}")
        if (
            recalc
            and kind == "middle"
            and coords_key != "zero"
            and not all(v is not None for v in values)
        ):
            coords = [
                self.get_m_cords(p)
                for p in (
                    self.p_last_coords[0],
                    self.p_last_coords[1],
                    self.p_last_coords[2],
                )
            ]
            if self.is_debug:
                print(f"[DEBUG] recalc coords ({kind}): {coords}")
        else:
            coords = [np.array(v) for v in values]
            if self.is_debug:
                print(f"[DEBUG] final coords ({kind}): {coords}")
        if not is_last_flag:
            self.update_last_coords(kind, *coords)
            setattr(self, f"{self.flag_map['coords'][kind]}", True)
        return tuple(coords)
    @staticmethod
    def segments_intersect(p1, p2, q1, q2, eps_segm=1e-9):
        """[пояснення вилучено для скороченого лістингу]"""
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
        def orient(a, b, c):
            return (b[0] - a[0]) * (c[1] - a[1]) - (b[1] - a[1]) * (c[0] - a[0])
        o1 = orient(p1, p2, q1)
        o2 = orient(p1, p2, q2)
        o3 = orient(q1, q2, p1)
        o4 = orient(q1, q2, p2)
        return o1 * o2 < -eps_segm and o3 * o4 < -eps_segm
    def is_valid_combo(self, B, B2, B3, combinations_p):
        """[пояснення вилучено для скороченого лістингу]"""
        C, C2, C3 = combinations_p
        segments = [(B, C[:2]), (B2, C2[:2]), (B3, C3[:2])]
        for i in range(len(segments)):
            for j in range(i + 1, len(segments)):
                if self.segments_intersect(
                    segments[i][0], segments[i][1], segments[j][0], segments[j][1]
                ):
                    return False
        return True
    def plot_combinations(self, combinations_m, combinations_p, decimals=2):
        """[пояснення вилучено для скороченого лістингу]"""
        if not combinations_m:
            print(" ⚠️ Немає валідних комбінацій")
            return
        C, C2, C3 = combinations_p
        def draw_point(ax, xy, *, color="blue", size=40, marker="o", label=None, z=3):
            """[пояснення вилучено для скороченого лістингу]"""
            ax.scatter(
                float(xy[0]),

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

                float(xy[1]),
                c=color,
                s=size,
                marker=marker,
                zorder=z,
                label=label,
            )
        def draw_line(ax, p1, p2, *, color="k", lw=1, ls="-", z=2):
            """[пояснення вилучено для скороченого лістингу]"""
            ax.plot(
                [float(p1[0]), float(p2[0])],
                [float(p1[1]), float(p2[1])],
                linestyle=ls,
                linewidth=lw,
                color=color,
                zorder=z,
            )
        def draw_label(
            ax,
            xy,
            text,
            *,
            fontsize=8,
            color="black",
            ha="left",
            va="bottom",
            offset=(0, 0),
        ):
            """[пояснення вилучено для скороченого лістингу]"""
            ax.text(
                float(xy[0]) + offset[0],
                float(xy[1]) + offset[1],
                text,
                fontsize=fontsize,
                color=color,
                ha=ha,
                va=va,
            )
        def xy(p):
            return (float(p[0]), float(p[1]))
        def dist3D(b, c):
            return float(np.linalg.norm([b[0] - c[0], b[1] - c[1], -c[2]]))
        cols = 4
        rows = (len(combinations_m) + cols - 1) // cols
        fig, axes = plt.subplots(rows, cols, figsize=(4 * cols, 4 * rows))
```

```
        axes = axes.flatten()
        for ax in axes:
            ax.add_patch(Circle((0, 0), self.R1, fill=False,
linestyle="--", alpha=0.6))
            ax.add_patch(Circle((0, 0), 40, fill=False,
linestyle="--", alpha=0.6))
        for i, (b, b2, b3) in enumerate(combinations_m):
            ax = axes[i]
            draw_point(ax, b, color="blue")
            draw_point(ax, b2, color="blue")
            draw_point(ax, b3, color="blue")
            draw_point(ax, xy(C), color="red")
            draw_point(ax, xy(C2), color="red")
            draw_point(ax, xy(C3), color="red")
            draw_point(ax, (0, 0), color="black", size=30)
            draw_line(ax, b, b2, color="blue")
            draw_line(ax, b2, b3, color="blue")
            draw_line(ax, b3, b, color="blue")
            draw_line(ax, xy(C), xy(C2), color="red", ls="-
-")
            draw_line(ax, xy(C2), xy(C3), color="red",
ls="--")
            draw_line(ax, xy(C3), xy(C), color="red", ls="-
-")
            for p in [b, b2, b3]:
                draw_line(ax, (0, 0), p, color="black", ls="--",
lw=1)
            # ── лінії: з'єднання пар B–C ──
            draw_line(ax, b, xy(C), color="green", lw=1)
            draw_line(ax, b2, xy(C2), color="green", lw=1)
            draw_line(ax, b3, xy(C3), color="green", lw=1)
            draw_label(
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
                ax,
                b,
                f"B({round(b[0], decimals)}; {round(b[1],
decimals)})",
                color="red",
                ha="right",
            )
            draw_label(
                ax,
                b2,
                f"B2({round(b2[0], decimals)};
{round(b2[1], decimals)})",
                color="green",
                ha="right",
            )
            draw_label(
                ax,
                b3,
                f"B3({round(b3[0], decimals)};
{round(b3[1], decimals)})",
                color="purple",
                ha="right",
            )
            draw_label(
                ax,
                xy(C),
                f"C({round(C[0], decimals)}; {round(C[1],
decimals)})",
                color="red",
            )
            draw_label(

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---


                ax,
                xy(C2),
                f"C2({round(C2[0], decimals)};
{round(C2[1], decimals)})",
                color="red",
            )
            draw_label(
                ax,
                xy(C3),
                f"C3({round(C3[0], decimals)};
{round(C3[1], decimals)})",
                color="red",
            )
            ax.set_title(f"Combo {i + 1}")
            ax.axhline(0, color="black", linewidth=0.5)
            ax.axvline(0, color="black", linewidth=0.5)
            ax.set_aspect("equal", adjustable="box")
        for j in range(len(combinations_m), len(axes)):
            fig.delaxes(axes[j])
        plt.tight_layout()
        plt.show()
    def filter_points_on_circle(self, points,
eps_circle=1e-6):
        """[пояснення вилучено для скороченого
лістингу]"""
        pts = np.array(points, dtype=float)
        mask = np.abs(np.sum(pts**2, axis=1) -
self.R1**2) < eps_circle
        pts = np.round(pts[mask], 8)
        pts = np.unique(pts, axis=0)
        return [tuple(p) for p in pts]
    @staticmethod
    def filter_valid_combinations(
        points, min_dist=10.0, eps_abs=1e-9,
round_decimals=8
    ):
        """[пояснення вилучено для скороченого
лістингу]"""
        B, B2, B3 = points
        def _unique_points(points):
            seen = set()
            out = []
            for x, y in points:
                key = (round(float(x), round_decimals),
round(float(y), round_decimals))
                if key not in seen:
                    seen.add(key)
                    out.append(key)
            return out
        B = _unique_points(B)
        B2 = _unique_points(B2)
        B3 = _unique_points(B3)
        md2 = float(min_dist) ** 2
        eps2 = float(eps_abs) ** 2
        thr2 = max(md2, eps2)
        valid_combinations = []
        seen_combos = set()
        for combo in itertools.product(B, B2, B3):
```

```
        ok = True
        for (x1, y1), (x2, y2) in itertools.combinations(combo, 2):
            dx, dy = x2 - x1, y2 - y1
            if dx * dx + dy * dy <= thr2:
                ok = False
                break
        if not ok:
            continue
        key = tuple(
            (round(x, round_decimals), round(y, round_decimals)) for (x, y) in combo
        )
        if key in seen_combos:
            continue
        seen_combos.add(key)
        valid_combinations.append(tuple((float(x), float(y)) for (x, y) in combo))
    return valid_combinations
  def get_config(self) -> tuple[dict, dict, dict, dict]:
    """[пояснення вилучено для скороченого лістингу]"""
    self.config = AnglesConfig(work_dirs=self.work_dirs, is_multi=self.is_multi)
    if self.is_debug:
        print(f"[DEBUG] Config loaded: {self.config.config_path}")
    self.all_p_coors = self.config.data["coordinates"]["platform"]
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
    self.all_m_coors = self.config.data["coordinates"]["middle"]
    self.all_p_angles = self.config.data["angles"]["platform"]
    self.all_s_angles = self.config.data["angles"]["servo"]
    return self.all_p_coors, self.all_m_coors, self.all_p_angles, self.all_s_angles
  def update_last_coords(self, coords_type, coords_1, coords_2, coords_3):
    """[пояснення вилучено для скороченого лістингу]"""
    mapping = {
        "platform": ["c", "c2", "c3"],
        "middle": ["b", "b2", "b3"],
    }
    if coords_type not in mapping:
        raise ValueError(f"Unknown coord_type '{coords_type}'")
    keys = mapping[coords_type]
    last_data = {
        keys[0]: coords_1.tolist(),
        keys[1]: coords_2.tolist(),
        keys[2]: coords_3.tolist(),
    }
    all_entry = [coords_1.tolist(), coords_2.tolist(), coords_3.tolist()]

self.config.update_value(f"coordinates.{coords_type}.last", last_data)
    cfg = self.config.data
cfg["coordinates"][coords_type]["all"].append(all_entry)
    self.config.update_value(
        f"coordinates.{coords_type}.all",
        cfg["coordinates"][coords_type]["all"],
    )

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

    self.config.load()
  def update_last_angles(
    self,
    angles_type: str,
    angles_1: float,
    angles_2: float,
    angles_3: float,
  ) -> None:
    """[пояснення вилучено для скороченого лістингу]"""
    mapping = {
        "platform": ["alpha", "beta", "gamma"],
        "servo": ["a", "a2", "a3"],
    }
    if angles_type not in mapping:
        raise ValueError(
            f"Unknown group '{angles_type}'. Очікується 'platform' або 'servo'."
        )
    keys = mapping[angles_type]
    last_data = {
        keys[0]: float(angles_1),
        keys[1]: float(angles_2),
        keys[2]: float(angles_3),
    }
    all_entry = [float(angles_1), float(angles_2), float(angles_3)]

self.config.update_value(f"angles.{angles_type}.last", last_data)
    cfg = self.config.data

cfg["angles"][angles_type]["all"].append(all_entry)
    self.config.update_value(
        f"angles.{angles_type}.all",
        cfg["angles"][angles_type]["all"],
    )
    self.config.load()
    setattr(self, f"is_{angles_type[0]}_last_angles", True)
  def get_p_coords(self, coords):
    return self._rotation_matrix() @ coords
  def _rotation_matrix(self):
    a, b, g = np.radians([self.alpha, self.beta, self.gamma])
    ca, cb, cg = np.cos([a, b, g])
    sa, sb, sg = np.sin([a, b, g])
    return np.array(
        [
            [cb * cg, cg * sa * sb - ca * sg, sa * sg + ca * cg * sb],
```

```
            [cb * sg, ca * cg + sa * sb * sg, ca * sb * sg - cg * sa],
            [sb, -cb * sa, ca * cb],
        ]
    )
  def get_m_cords(self, m):
    x, y, z = float(m[0]), float(m[1]), float(m[2])
    if z * z > self.Rc * self.Rc:
      if self.is_debug:
        print(
          f"⚠️ Неприпустиме z: z^2={z * z:.6f} > Rc^2={self.Rc * self.Rc:.6f}"
        )
      return None
    r2 = np.sqrt(self.Rc**2 - z**2)
    A = -(self.R1**2) + 2 * self.R1 * r2 - r2**2 + x**2 + y**2
    B = self.R1**2 + 2 * self.R1 * r2 + r2**2 - x**2 - y**2
    under_sqrt = A * B
    if under_sqrt < 0:
      if self.is_debug:
        print(
          "⚠️ Уявні координати (підкореневий вираз < 0). "
          f"alpha={self.alpha:.6f}, beta={self.beta:.6f}, gamma={self.gamma:.6f}; "
          f"m=({x:.6f},{y:.6f},{z:.6f}); under_sqrt={under_sqrt:.6e}"
        )
      return None
    f = np.sqrt(under_sqrt)
    denom1 = 2 * x * (x**2 + y**2)
    denom2 = 2 * (x**2 + y**2)
    tiny = 1e-12
    if abs(denom1) < tiny or abs(denom2) < tiny:
      if self.is_debug:
        print(
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
          "⚠️ Вироджена геометрія (ділення дуже мале). "
          f"denom1={denom1:.6e}, denom2={denom2:.6e}, m=({x},{y})"
        )
      return None
    bx1 = (
      y * (x * f - y * (self.R1**2 - r2**2 + x**2 + y**2))
      + (x**2 + y**2) * (self.R1**2 - r2**2 + x**2 + y**2)
    ) / denom1
    bx2 = (
      -y * (x * f + y * (self.R1**2 - r2**2 + x**2 + y**2))
      + (x**2 + y**2) * (self.R1**2 - r2**2 + x**2 + y**2)
    ) / denom1
    by1 = (-x * f + y * (self.R1**2 - r2**2 + x**2 + y**2)) / denom2
    by2 = (x * f + y * (self.R1**2 - r2**2 + x**2 + y**2)) / denom2
    return float(bx1), float(bx2), float(by1), float(by2)

  def filter_by_bend_side(self, combos, combinations_p):
    """[пояснення вилучено для скороченого лістингу]"""
    if not combos:
      return []
    C, C2, C3 = combinations_p
    def vector_angle_sign(C, B):
      """[пояснення вилучено для скороченого лістингу]"""
      vC = np.array([C[0], C[1]])
      vB = np.array([B[0], B[1]])
      cross = vC[0] * vB[1] - vC[1] * vB[0]

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

      return np.sign(cross)
    def combo_bend_sign(combo):
      """[пояснення вилучено для скороченого лістингу]"""
      (B, B2, B3) = combo
      signs = [
        vector_angle_sign(C[:2], B),
        vector_angle_sign(C2[:2], B2),
        vector_angle_sign(C3[:2], B3),
      ]
      return np.sign(sum(signs))
    desired_sign = -1 if self.is_bend else +1
    filtered = [combo for combo in combos if combo_bend_sign(combo) == desired_sign]
    if not filtered and self.is_debug:
      print(
        f"⚠️ Всі {len(combos)} комбінацій не збігаються з is_bend={self.is_bend}. "
        f"Повертаємо весь набір без фільтрації."
      )
      return combos
    return filtered
  def get_angles(
    self,
    m_last_coords,
    valid_combos: list[
      tuple[tuple[float, float], tuple[float, float], tuple[float, float]]
    ],
  ) -> tuple[float, float, float]:
    """[пояснення вилучено для скороченого лістингу]"""
    m_last_coords_1, m_last_coords_2, m_last_coords_3 = m_last_coords
    b_vectors = MathFunction.get_vector(
      m_last_coords_1[0],
      m_last_coords_1[1],
      valid_combos[0][0][0],
      valid_combos[0][0][1],
    )
    b2_vectors = MathFunction.get_vector(
      m_last_coords_2[0],
      m_last_coords_2[1],
      valid_combos[0][1][0],
      valid_combos[0][1][1],
    )
```

```
        b3_vectors = MathFunction.get_vector(
            m_last_coords_3[0],
            m_last_coords_3[1],
            valid_combos[0][2][0],
            valid_combos[0][2][1],
        )
        a_sin = MathFunction.sin_alpha(b_vectors, self.R1)
        a2_sin = MathFunction.sin_alpha(b2_vectors, self.R1)
        a3_sin = MathFunction.sin_alpha(b3_vectors, self.R1)
        if self.is_debug:
            print(
                f"{self.m_last_coords[0]}, {m_last_coords_1[1]}, {valid_combos[0][0][0]}, {valid_combos[0][0][1]}\n"
                f"Вектор для точки B: {b_vectors}"
            )
            print(
                f"{m_last_coords_2[0]}, {m_last_coords_2[1]}, {valid_combos[0][1][0]}, {valid_combos[0][1][1]}\n"
                f"Вектор для точки B2: {b2_vectors}"
            )
            print(
                f"{m_last_coords_3[0]}, {m_last_coords_3[1]}, {valid_combos[0][2][0]}, {valid_combos[0][2][1]}\n"
                f"Вектор для точки B3: {b3_vectors}"
            )
            print(f"Синуси кутів: A - {a_sin}, A2 - {a2_sin}, A3 - {a3_sin}")
        a = 180 / np.pi * arcsin(a_sin)
        a2 = 180 / np.pi * arcsin(a2_sin)
        a3 = 180 / np.pi * arcsin(a3_sin)
        return a, a2, a3
    @staticmethod
    def determ_num(
        *,
        m_last_coords: tuple[float, float],
        valid_combos: list[
            tuple[tuple[float, float], tuple[float, float], tuple[float, float]]
        ],
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
        k: int,
    ) -> tuple[int, int]:
        """[пояснення вилучено для скороченого лістингу]"""
        det_x = valid_combos[0][k][0] - m_last_coords[0]
        det_y = valid_combos[0][k][1] - m_last_coords[1]
        sign = lambda v: int(v > 0) - int(v < 0)
        return sign(det_x), sign(det_y)
    @staticmethod
    def build_parser():
        epilog_text = """[пояснення вилучено для скороченого лістингу]"""
        parser = argparse.ArgumentParser(
            prog="get_angles",
            description="Обчислення кутів сервоприводів (A, A2, A3) для заданих α, β, γ.",
formatter_class=argparse.RawDescriptionHelpFormatter,
            epilog=epilog_text,
        )
        parser.add_argument("alpha", type=float, help="Кут α (в градусах)")
        parser.add_argument("beta", type=float, help="Кут β (в градусах)")
        parser.add_argument("gamma", type=float, help="Кут γ (в градусах)")
        parser.add_argument(
            "--is_bend",
            action=argparse.BooleanOptionalAction,
            default=True,
            help="Вибір сторони згину: True — права, False — ліва",

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        )
        parser.add_argument(
            "--is_debug",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="[DEBUG_INFO] Режим налагодження (додаткові повідомлення). (default: False)",
        )
        parser.add_argument(
            "--is_multi",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Багатократний режим (для пакетної обробки). (default: False)",
        )
        parser.add_argument(
            "--rc",
            type=float,
            default=75.0,
            help="Радіус Rc",
        )
        parser.add_argument(
            "--r1",
            type=float,
            default=50.0,
            help="Радіус R1",
        )
        parser.add_argument(
            "--is_2d",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Побудова 2D графіків",
        )
        parser.add_argument(
            "--is_3d",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="Побудова 3D графіків",
        )
        return parser
    def run_terminal(self):
        parser = self.build_parser()
```

```
        if len(sys.argv) == 1:
            parser.print_help()
            sys.exit(0)
        args = parser.parse_args()
        app = GetAngles(
            is_bend=args.is_bend,
            is_debug=args.is_debug,
            is_multi=args.is_multi,
            rc=args.rc,
            r1=args.r1,
            is_2d=args.is_2d,
            is_3d=args.is_3d,
        )
        result = app.calculation(alpha=args.alpha,
beta=args.beta, gamma=args.gamma)
        print(result)
        return result
    def run(self):
        result = self.calculation(
            alpha=self.alpha,
            beta=self.beta,
            gamma=self.gamma,
        )
        if self.is_debug:
            print(f"[OK] Run finished. Servo angles:
{result}")
        return result
if __name__ == "__main__":
    app = GetAngles()
    if len(sys.argv) > 1:
        app.run_terminal()
    else:
        print("[INFO] Running GetAngles in default mode
(no CLI args).")
        app.run()
"""[пояснення вилучено для скороченого
лістингу]"""
import os
import argparse
import json
import sys
import numpy as np
import trimesh
import matplotlib.pyplot as plt
from pathlib import Path
from common.project_manager import
get_default_work_dirs
class ObjToAngles:
    def __init__(
        self,
        input_file: str | None = None,
        output_file: str | None = None,
        filtered_file: str | None = None,
        *,
        unit: str = "deg",
        unique: bool = True,
        sample: int = 0,
        line_samples: int = 50,
        is_filter: bool = False,
        is_clear: bool = False,
        is_debug: bool = False,
    ):
        """[пояснення вилучено для скороченого
лістингу]"""
        self.work_dirs = get_default_work_dirs()
raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

        self.obj_dir = self.work_dirs.obj_dir
        self.json_dir = self.work_dirs.json_dir
        defaults = {
            "input": ("input.obj", self.obj_dir),
            "output": ("output.json", self.json_dir),
            "filtered": ("filtered.json", self.json_dir),
        }
        def resolve_path(filename: str | None, key: str):
            default_name, base_dir = defaults[key]
            base_dir = Path(base_dir)
            if filename:
                return filename if os.path.isabs(filename) else
base_dir / filename
            return base_dir / default_name
        self.input_file = resolve_path(input_file, "input")
        self.output_file = resolve_path(output_file,
"output")
        self.filtered_file = resolve_path(filtered_file,
"filtered")
        self.unit = unit
        self.unique = unique
        self.sample = sample
        self.line_samples = line_samples
        self.is_filter = is_filter
        self.is_clear = is_clear
        self.is_debug = is_debug
        self.selected_indices = set()
    @staticmethod
    def compute_angles(point, unit="deg"):
        x, y, z = point
        alpha = np.arctan2(y, z)
        beta = np.arctan2(x, z)
        gamma = np.arctan2(y, x)
        if unit == "deg":
            alpha = np.degrees(alpha)
            beta = np.degrees(beta)
            gamma = np.degrees(gamma)
        return float(alpha), float(beta), float(gamma)
    @staticmethod
    def unique_rows_preserve_order(arr):
        seen = {}
        unique = []
        for row in arr:
            key = tuple(map(float, row))
            if key not in seen:
                seen[key] = True
                unique.append(row)
        return np.array(unique, dtype=float)
    @staticmethod
    def discretize_lines(mesh, line_samples=50):
        points = []
        if hasattr(mesh, "entities"):
            for ent in mesh.entities:
                if
ent.__class__.__name__.lower().startswith("line"):
                    verts = mesh.vertices[ent.points]
                    for i in range(len(verts) - 1):
                        p0, p1 = verts[i], verts[i + 1]
                        t = np.linspace(0, 1, line_samples)
```

```
                seg = np.outer(1 - t, p0) + np.outer(t, p1)
                points.extend(seg)
        return np.array(points)
    def visualize_and_select_from_file(self, input_json, output_json="filtered.json"):
        with open(input_json, "r", encoding="utf-8") as f:
            data = json.load(f)
        points = data["points"]
        angles = data["angles"]
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        xs = [p["x"] for p in points]
        ys = [p["y"] for p in points]
        zs = [p["z"] for p in points]
        ax.scatter(xs, ys, zs, c="blue", picker=True, s=30)
        def on_pick(event):
            ind = event.ind[0]
            if ind in self.selected_indices:
                self.selected_indices.remove(ind)
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
                ax.scatter(xs[ind], ys[ind], zs[ind], c="blue", s=50)
            else:
                self.selected_indices.add(ind)
                ax.scatter(xs[ind], ys[ind], zs[ind], c="red", s=80, marker="o")
            plt.draw()
        def on_key(event):
            if event.key.lower() in {"e", "y"}:
                filtered_points = [points[i] for i in self.selected_indices]
                filtered_angles = [angles[i] for i in self.selected_indices]
                output_data = {"points": filtered_points, "angles": filtered_angles}
                with open(output_json, "w", encoding="utf-8") as f_out:
                    json.dump(
                        output_data,
                        f_out,
                        ensure_ascii=False,
                        indent=2,
                    )
                print(f"[OK] Saved {len(filtered_points)} points -> {output_json}")
        fig.canvas.mpl_connect("pick_event", on_pick)
        fig.canvas.mpl_connect("key_press_event", on_key)
        plt.title("Клік по точках = вибір, 'E' = зберегти (export)")
        plt.show()
    def process_mesh(
        self,
        input_file: str | None = None,


raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        output_file: str | None = None,
        filtered_file: str | None = None,
        *,
        unit: str | None = None,
        unique: bool | None = None,
        sample: int | None = None,
        line_samples: int | None = None,
        is_filter: bool | None = None,
        is_clear: bool | None = None,
        is_debug: bool | None = None,
    ):
        """[пояснення вилучено для скороченого лістингу]"""
        input_file = input_file or self.input_file
        output_file = output_file or self.output_file
        filtered_file = filtered_file or self.filtered_file
        unit = unit or self.unit
        unique = self.unique if unique is None else unique
        sample = self.sample if sample is None else sample
        line_samples = self.line_samples if line_samples is None else line_samples
        is_filter = self.is_filter if is_filter is None else is_filter
        is_clear = self.is_clear if is_clear is None else is_clear
        is_debug = self.is_debug if is_debug is None else is_debug
        scene_or_mesh = trimesh.load(input_file)
        if isinstance(scene_or_mesh, trimesh.Scene):
            meshes = list(scene_or_mesh.geometry.values())
            if not meshes and is_debug:
                print("❌ У сцені немає геометрії")
                sys.exit(2)
            mesh = trimesh.util.concatenate(meshes)
        else:
            mesh = scene_or_mesh
        verts = np.asarray(mesh.vertices)
        line_points = self.discretize_lines(mesh, line_samples)
        if line_points.size > 0:
            verts = np.vstack((verts, line_points))
        if verts.size == 0 and is_debug:
            print("❌ В obj немає вершин або ліній.")
            sys.exit(2)
        if unique:
            verts = self.unique_rows_preserve_order(verts)
        if sample and 0 < sample < len(verts):
            rng = np.random.default_rng(0)
            idx = rng.choice(len(verts), size=sample, replace=False)
            verts = verts[np.sort(idx)]
        result = {"points": [], "angles": []}
        for v in verts:
            alpha, beta, gamma = self.compute_angles(v, unit=unit)
            x, y, z = v
            y, z = z, y
            result["points"].append({"x": x, "y": y, "z": z})
            result["angles"].append({"alpha": alpha, "beta": beta, "gamma": gamma})
        with open(output_file, "w", encoding="utf-8") as f:
            json.dump(result, f, ensure_ascii=False, indent=2)
        if is_debug:
            print(f"[OK] Saved {len(result['points'])} points → {output_file}")
        if is_filter:
```

```
        self.visualize_and_select_from_file(output_file, filtered_file)
        if is_clear and os.path.exists(output_file):
            os.remove(output_file)
    @staticmethod
    def build_parser():
        epilog_text = """[пояснення вилучено для скороченого лістингу]"""
        parser = argparse.ArgumentParser(
            prog="obj_to_angles",
            description="Convert OBJ → JSON with calculated angles (α, β, γ) for each vertex or line point.",
            formatter_class=argparse.RawDescriptionHelpFormatter,
            epilog=epilog_text,
        )
        parser.add_argument("input", help="input OBJ file")
        parser.add_argument("output", help="output JSON file")
        parser.add_argument(
            "--unit",
            choices=["deg", "rad"],
            default="deg",
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
            help="units for angles (default: deg)",
        )
        parser.add_argument(
            "--unique-vertices",
            dest="unique",
            action=argparse.BooleanOptionalAction,
            default=True,
            help="remove duplicate vertices (default: True) or keep all vertices, even duplicates (False)",
        )
        parser.add_argument(
            "--sample",
            type=int,
            default=0,
            help="random sample of N points (0 = all, default: 0)",
        )
        parser.add_argument(
            "--line-samples",
            type=int,
            default=50,
            help="number of points per line segment (default: 50)",
        )
        parser.add_argument(
            "--filter",
            action=argparse.BooleanOptionalAction,

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            default=False,
            help="enable the point sampling function (saves to filtered.json, default: False)",
        )
        parser.add_argument(
            "--clear",
            action=argparse.BooleanOptionalAction,
            default=False,
            help="delete intermediate file 'output.json' (default: False)",
        )
        parser.add_argument(
            "--is_debug",
            action=argparse.BooleanOptionalAction,
            default=False,
            help=" [DEBUG_INFO] (default: False)",
        )
        return parser
    def run_terminal(self):
        parser = self.build_parser()
        if len(sys.argv) == 1:
            parser.print_help()
            sys.exit(0)
        args = parser.parse_args()
        self.process_mesh(
            args.input,
            args.output,
            unit=args.unit,
            unique=args.unique,
            sample=args.sample,
            line_samples=args.line_samples,
            is_filter=args.filter,
            is_clear=args.clear,
            is_debug=args.is_debug,
        )
    def run(self):
        self.process_mesh(
            self.input_file,
            self.output_file,
            self.filtered_file,
            unit=self.unit,
            unique=self.unique,
            sample=self.sample,
            line_samples=self.line_samples,
            is_filter=self.is_filter,
            is_clear=self.is_clear,
            is_debug=self.is_debug,
        )
if __name__ == "__main__":
    app = ObjToAngles()
    if len(sys.argv) > 1:
        app.run_terminal()
    else:
        print("[INFO] Running ObjToAngles in default mode (no CLI args).")
        app.run()

#
# === project_listing_redacted.py
#

import threading
import subprocess
import sys, os, runpy, json, shutil
from pathlib import Path
```

```
from common.project_manager import WorkDirs, ConfigManager
class ApiConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.3
    def __init__(self, config_name: str='api_config.json', work_dirs=None, template_name: str='api_config.json.default'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _save_default_config_template(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_config = self._get_default_structure()
        with open(self.template_path, 'w', encoding='utf-8') as f:
            json.dump(default_config, f, ensure_ascii=False, indent=4)
        print(f'[OK] Created default API template: {self.template_path}')
    def ensure_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def validate_structure(self):
        required_keys = {'server', 'paths', 'modules', 'ui', 'debug'}
        if not isinstance(self.data, dict):
            return False
        return required_keys.issubset(self.data.keys())
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def update_value(self, path_str: str, new_value, autosave: bool=True):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def server(self):
        return self.data.get('server', {})
    def paths(self):
        return self.data.get('paths', {})
    def extensions(self):
        return self.data.get('extensions')
    def modules(self):
        return self.data.get('modules', {})
    def visualization(self):
        return self.data.get('visualization', {})
    def debug(self):
        return self.data.get('debug', {})
    def ui(self):
        return self.data.get('ui', {})
    def work_with_ip(self):
        return self.data.get('ip', {})
class ProcessManager:
    """[пояснення вилучено для скороченого лістингу]"""
    def __init__(self, work_dirs: WorkDirs | None=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _get_my_work_dir(self):
        return self.work_dirs.__repr__()
    def start_process(self, name: str, command: list[str]) -> None:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def check_status(self, name: str) -> dict[str, str | None] | str:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def stop_process(self, name: str) -> None:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run_module(self, name: str, args: dict | None=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run_all_enabled_modules(self):
        for name, module in self.modules_config.items():
            if module.get('enabled', False):
                self.run_module(name)
    def cleanup(self):
        """[пояснення вилучено для скороченого лістингу]"""
        for name, process in self.active_processes.items():
            if process.poll() is None:
                process.terminate()
                if self.is_debug:
                    print(f"[DEBUG] Terminated process '{name}' (PID {process.pid}).")
    def shutdown(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_status_report(self) -> dict:
        report = {}
        for name, process in self.active_processes.items():
            status = 'running' if process.poll() is None else 'finished'
            report[name] = {'pid': process.pid, 'status': status}
        return report
    def check_ip_access(self, request) -> bool:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
import os, sys, threading, time
from datetime import datetime
import socket
from fastapi import FastAPI, HTTPException, Request, UploadFile, File
from fastapi.responses import JSONResponse
import uvicorn
from pathlib import Path
from api.process_manager import ProcessManager
```

```
raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

from common.project_manager import WorkDirs
def _set_my_work_dir():
    """[пояснення вилучено для скороченого лістингу]"""
    pass
base_dir = _set_my_work_dir()
work_dirs = WorkDirs(base_dir=base_dir)
manager = ProcessManager(work_dirs=work_dirs)
app = FastAPI(title='SPM API Server', description='API для керування модулями SPM-системи.', version='0.2', docs_url='/__dev_docs', redoc_url='/__redoc_open_docs', openapi_url='/__openapi_docs.json')
@app.middleware('http')
async def protect_docs(request: Request, call_next):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.post('/__get_my_work_dir', include_in_schema=False)
def _get_my_work_dir(request: Request):
    if not manager.check_ip_access(request):
        raise HTTPException(status_code=403, detail='Access denied (IP restricted)')
    prog_dir = manager._get_my_work_dir()
    return {prog_dir}
@app.post('/api/__testing_is_running', include_in_schema=False)
def test():
    """[пояснення вилучено для скороченого лістингу]"""
    return {'status': 'ok', 'message': 'SPM API is running'}
@app.post('/api/__shutdown', include_in_schema=False)
def shutdown(request: Request):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.get('/api/ping')
async def ping():
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.post('/api/upload')
async def upload_file(file: UploadFile=File(...)):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.get('/api/modules')
def list_modules():
    """[пояснення вилучено для скороченого лістингу]"""
    return manager.modules_config
@app.get('/api/status')
def get_status():
    """[пояснення вилучено для скороченого лістингу]"""
    report = manager.get_status_report()
    return JSONResponse(content=report)
@app.post('/api/start/{module_name}')
async def start_module(module_name: str, request: Request):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.post('/api/stop/{module_name}')
def stop_module(module_name: str):
    """[пояснення вилучено для скороченого лістингу]"""
    manager.stop_process(module_name)
    return {'status': 'stopped', 'module': module_name}
@app.post('/api/restart/{module_name}')
def restart_module(module_name: str):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
@app.post('/api/start_all', include_in_schema=False)
def start_all():
    """[пояснення вилучено для скороченого лістингу]"""
    manager.run_all_enabled_modules()
    return {'status': 'started_all'}
@app.post('/api/stop_all')
def stop_all():
    """[пояснення вилучено для скороченого лістингу]"""
    manager.shutdown()
    return {'status': 'all stopped'}
@app.get('/api/config')
def get_config():
    """[пояснення вилучено для скороченого лістингу]"""
    return JSONResponse(content=manager.config.data)
if __name__ == '__main__':
    raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
import sys, os, json
import tempfile
from pathlib import Path
def get_default_work_dirs(is_debug: bool=False):
    """[пояснення вилучено для скороченого лістингу]"""
    pass
class WorkDirs:
    def __init__(self, base_dir: str, is_debug: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def __repr__(self):
        return f'WorkDirs(base={self.base_dir})'
class ConfigManager:
    """[пояснення вилучено для скороченого лістингу]"""
    def __init__(self, config_name: str, base_dir: str='json'):
        self.data = {}
        self.config_base_dir = Path(base_dir)
        self.config_base_dir.mkdir(parents=True, exist_ok=True)
        self.config_path = self.config_base_dir / config_name
        if self.config_path.exists():
            self.load()
```

```
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        with open(self.config_path, 'r', encoding='utf-8') as f:
            self.data = json.load(f)
    def save(self):
        """[пояснення вилучено для скороченого лістингу]"""

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        with open(self.config_path, 'w', encoding='utf-8') as f:
            json.dump(self.data, f, ensure_ascii=False, indent=4)
    def get(self, *keys, default=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def set(self, *keys, value):
        """[пояснення вилучено для скороченого лістингу]"""
        node = self.data
        for key in keys[:-1]:
            node = node.setdefault(key, {})
        node[keys[-1]] = value
    def __repr__(self):
        return f'ConfigManager({self.config_path.name}, keys={len(self.data)})'
import argparse, sys, json
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
from pathlib import Path
from concurrent.futures import ProcessPoolExecutor, as_completed
from getangles.getangles import GetAngles
from objtoangles.objtoangles import ObjToAngles
from common.project_manager import ConfigManager, get_default_work_dirs
class DetermStatic:
    @staticmethod
    def _generate_new_path(output_file=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def process_angle(point, angle, is_debug=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def filter_and_visualize_points(input_json='filtered.json', output_json='valid_result.json'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
class AreaConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.1
    def __init__(self, config_name='area_config.json', work_dirs=None, is_debug=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def ensure_default_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def load_data(path):
        """[пояснення вилучено для скороченого лістингу]"""
        path = Path(path)
        if not path.exists():
            raise FileNotFoundError(f'JSON file not found: {path}')
        with open(path, 'r', encoding='utf-8') as f:
            return json.load(f)
    @staticmethod
    def save_json(path, data):
        """[пояснення вилучено для скороченого лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, 'w', encoding='utf-8') as f:
            json.dump(data, f, ensure_ascii=False, indent=2)
        print(f'[OK] Saved JSON → {path}')
    @staticmethod
    def save_obj(path, vertices):
        """[пояснення вилучено для скороченого лістингу]"""
        path = Path(path)
        path.parent.mkdir(parents=True, exist_ok=True)
        with open(path, 'w', encoding='utf-8') as f:
            f.write('\n'.join(vertices))
        print(f'[OK] Saved OBJ ({len(vertices)} points) → {path}')
class DetermWorkArea:
    def __init__(self, input_obj: str='points.obj', conv_out: str='conv_output.json', valid_out: str='valid.json', valid_res: str='valid_result.json', output: str='output.json', unoutput: str='unoutput.json', *, gen_new_path: bool=False, is_new_path: bool=False, is_validate: bool=True, is_simpify: bool=False, is_check: bool=True, is_debug: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _convert_obj_to_angles(self):
        converter = ObjToAngles(self.input_obj_path, self.conv_out_path, is_filter=False)
        converter.run()
    def _validate_angles(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def check_perm_angles(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def build_parser():
```

```
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def run_terminal():
        """[пояснення вилучено для скороченого лістингу]"""
        pass
if __name__ == '__main__':
    raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
"""[пояснення вилучено для скороченого лістингу]"""
import os, sys, re, json, shutil
import argparse
import numpy as np
from numpy import arcsin
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
import itertools

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

from mpl_toolkits.mplot3d import Axes3D
from pathlib import Path
from common.project_manager import ConfigManager, get_default_work_dirs
'\nСловник скорочень:\nS, s - servo - сервоприводи або шарніри A, A2, A3\nP, p - platform - платформа SPM - C, C2, C3,\nM, m - middle - шарніри B, B2, B3\n'
class MathFunction:
    @staticmethod
    def sin_alpha(a, b):
        """[пояснення вилучено для скороченого лістингу]"""
        return a / b ** 2 * np.sqrt(b ** 2 - a ** 2 / 4)
    @staticmethod
    def get_vector(bx0, by0, bx, by):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
class AnglesConfig(ConfigManager):
    """[пояснення вилучено для скороченого лістингу]"""
    CURRENT_VERSION = 0.1
    def __init__(self, config_name='angles_config.json', work_dirs=None,
template_name='angles_config.json.default', is_multi: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _save_default_config_template(self):
        """[пояснення вилучено для скороченого лістингу]"""
        default_config = self._get_default_structure()
        with open(self.template_path, 'w', encoding='utf-8') as f:
            json.dump(default_config, f, ensure_ascii=False, indent=4)
        if self.work_dirs and self.work_dirs.is_debug:
            print(f'[DEBUG] Created default config: {self.config_path}')
    def ensure_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def validate_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def load(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def create_instance_config(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def update_value(self, path_str, new_value):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _get_default_structure(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
class GetAngles:
    def __init__(self, *, is_bend: bool=True, is_debug: bool=False, is_multi: bool=False, rc: float=75.0, r1: float=50.0, is_2d: bool=False, is_3d: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def calculation(self, alpha=None, beta=None, gamma=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def p_absolute_angles(self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode='delta'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def p_angles(self, p_last_angles_1, p_last_angles_2, p_last_angles_3, *, mode='delta'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def s_angles(s_last_angles_1, s_last_angles_2, s_last_angles_3, dir_b, dir_b2, dir_b3, a, a2, a3):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def plot_mechanism_3d(self, p_last_coords, m_last_coords, p_coords, a: float, a2: float, a3: float, *, valid_combos: list | None=None, title: str='3D Visualization', axis_range: float=120):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _get_last_angles(self, angles_type: str) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
```

```
        pass
    def get_p_last_angles(self) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        return self._get_last_angles('platform')
    def get_s_last_angles(self) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        return self._get_last_angles('servo')
    def circle_direction(self, m_last_coords, valid_combos, k):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def choose_closest_combo(combos, last_coords):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_p_last_coords(self):
        return self._get_last_coords(kind='platform', recalc=False)
    def get_m_last_coords(self):
        return self._get_last_coords(kind='middle', recalc=False)
    def _get_last_coords(self, *, kind: str, recalc: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def segments_intersect(p1, p2, q1, q2, eps_segm=1e-09):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def is_valid_combo(self, B, B2, B3, combinations_p):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def plot_combinations(self, combinations_m, combinations_p, decimals=2):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def filter_points_on_circle(self, points, eps_circle=1e-06):
        """[пояснення вилучено для скороченого лістингу]"""

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        pass
    @staticmethod
    def filter_valid_combinations(points, min_dist=10.0, eps_abs=1e-09, round_decimals=8):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_config(self) -> tuple[dict, dict, dict, dict]:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def update_last_coords(self, coords_type, coords_1, coords_2, coords_3):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def update_last_angles(self, angles_type: str, angles_1: float, angles_2: float, angles_3: float) -> None:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_p_coords(self, coords):
        return self._rotation_matrix() @ coords
    def _rotation_matrix(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_m_cords(self, m):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def filter_by_bend_side(self, combos, combinations_p):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def get_angles(self, m_last_coords, valid_combos: list[tuple[tuple[float, float], tuple[float, float], tuple[float, float]]]) -> tuple[float, float, float]:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def determ_num(*, m_last_coords: tuple[float, float], valid_combos: list[tuple[tuple[float, float], tuple[float, float], tuple[float, float]]], k: int) -> tuple[int, int]:
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def build_parser():
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run_terminal(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
if __name__ == '__main__':
    raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
"""[пояснення вилучено для скороченого лістингу]"""
import os
import argparse
import json
import sys
```

```
import numpy as np
import trimesh
import matplotlib.pyplot as plt
from pathlib import Path
from common.project_manager import get_default_work_dirs
class ObjToAngles:
    def __init__(self, input_file: str | None=None, output_file: str | None=None, filtered_file: str | None=None, *, unit: str='deg', unique: bool=True, sample: int=0, line_samples: int=50, is_filter: bool=False, is_clear: bool=False, is_debug: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def compute_angles(point, unit='deg'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def unique_rows_preserve_order(arr):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def discretize_lines(mesh, line_samples=50):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def visualize_and_select_from_file(self, input_json, output_json='filtered.json'):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def process_mesh(self, input_file: str | None=None, output_file: str | None=None, filtered_file: str | None=None, *, unit: str | None=None, unique: bool | None=None, sample: int | None=None, line_samples: int | None=None, is_filter: bool | None=None, is_clear: bool | None=None, is_debug: bool | None=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    @staticmethod
    def build_parser():
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run_terminal(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def run(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
if __name__ == '__main__':
    raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
import cv2
import mediapipe as mp
import numpy as np
import math
import time
from getangles.getangles import GetAngles
class HandToPlatformController:
    def __init__(self, camera_index=0, smoothing=0.8, delay=2.5, scale_factors=(1.0, 1.0, 1.0), limits=((-46, 46), (-46, 46), (-180, 180))):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _calculate_angles(self, landmarks):

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _smooth_angles(self, new_angles):
        self.prev_angles = self.smoothing * self.prev_angles + (1 - self.smoothing) * new_angles
        return self.prev_angles
    def _adjust_for_hand(self, angles):
        if self.hand_label == 'Left':
            angles[0] *= -1
            angles[2] *= -1
        return angles
    def _calibrate(self, current_angles):
        self.zero_angles = current_angles.copy()
        self.calibrated = True
        print('✅ Калібрування виконано. Поточне положення прийнято за нульове.')
    def _apply_scale_and_limits(self, angles):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def process_frame(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def release(self):
        self.cap.release()
        cv2.destroyAllWindows()
is_debug = False
controller = HandToPlatformController(delay=0.5)
while True:
    frame, angles = controller.process_frame()
    if frame is None:
        break
    if angles is not None:
        alpha, beta, gamma = angles
        print(f'Кути руки (α, β, γ): {alpha:.1f}, {beta:.1f}, {gamma:.1f}')
        servo_angles = GetAngles(is_debug=is_debug, is_multi=True).calculation(alpha=alpha, beta=beta, gamma=gamma)
        if all((not np.isnan(angle) for angle in servo_angles)):
            print(f's1: {servo_angles[0]}, s2: {servo_angles[1]}, s3: {servo_angles[2]}')
    cv2.imshow('Hand Tracking', frame)
    if cv2.waitKey(1) & 255 == 27:
        break
```

```
controller.release()
import cv2
import mediapipe as mp
import numpy as np
import math
import time
class HandToPlatformController:
    def __init__(self, camera_index=0, smoothing=0.8, delay=2.5):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _calculate_angles(self, landmarks):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def _smooth_angles(self, new_angles):
        self.prev_angles = self.smoothing * self.prev_angles + (1 - self.smoothing) * new_angles
        return self.prev_angles
    def process_frame(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def release(self):
        self.cap.release()
        cv2.destroyAllWindows()
controller = HandToPlatformController()
while True:
    frame, angles = controller.process_frame()
    if frame is None:
        break
    if angles is not None:
        print(f'Кути руки (roll, pitch, yaw): {angles}')
    cv2.imshow('Hand Tracking', frame)
    if cv2.waitKey(1) & 255 == 27:
        break
controller.release()
import serial.tools.list_ports
def _get_com_ports():
    ports = serial.tools.list_ports.comports()
    return [port.device for port in ports]
import serial
import time
from typing import Optional, Dict
class WorkWithArduino:
    def __init__(self, angles: Optional[tuple[float, float, float]]=None, message: Optional[str]=None, port: str='/dev/ttyUSB0', baudrate: int=9600, is_debug: bool=False, timeout: float=1.0, is_inf: bool=False):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def connect(self):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def send(self, angles: Optional[Dict[str, float]]=None, message: Optional[str]=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass
    def receive(self):
        """[пояснення вилучено для скороченого лістингу]"""
        if self.ser and self.ser.in_waiting > 0:
            data = self.ser.readline().decode().strip()
            print(f'📨 Отримано: {data}')
            return data
        return None
    def close(self):
        """[пояснення вилучено для скороченого лістингу]"""
        if self.ser and self.ser.is_open:
            self.ser.close()

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

            if self.is_debug:
                print(f'[DEBUG] Порт {self.port} закрито.')
    def _format_angles(self, angles: Dict[str, float]) -> str:
        """[пояснення вилучено для скороченого лістингу]"""
        if not angles:
            raise ValueError('Словник angles не може бути порожнім')
        formatted = ';'.join((str(angles.get(k, 0)) for k in ('s1', 's2', 's3'))) + '|'
        if self.is_debug:
            print(f'[DEBUG] Форматовані кути: {formatted}')
        return formatted
    def main(self, angles: Dict[str, float]=None, message: str=None):
        """[пояснення вилучено для скороченого лістингу]"""
        pass

#
# ===== rtt_controllers/ai_rtt.py
#

import cv2
import mediapipe as mp
import numpy as np
import math
import time
from getangles.getangles import GetAngles
class HandToPlatformController:
    def __init__(
        self,
        camera_index=0,
        smoothing=0.8,
        delay=2.5,
        scale_factors=(1.0, 1.0, 1.0),
        limits=((-46, 46), (-46, 46), (-180, 180)),
    ):
        self.cap = cv2.VideoCapture(camera_index)
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(max_num_hands=1)
        self.mp_draw = mp.solutions.drawing_utils
```

```
        self.prev_angles = np.zeros(3)
        self.smoothing = smoothing
        self.delay = delay
        self.last_time = 0
        self.scale_factors = np.array(scale_factors)
        self.limits = limits
        self.calibrated = False
        self.zero_angles = np.zeros(3)
        self.hand_label = None
    def _calculate_angles(self, landmarks):
        wrist = np.array([landmarks[0].x, landmarks[0].y, landmarks[0].z])
        middle = np.array([landmarks[9].x, landmarks[9].y, landmarks[9].z])
        index_tip = np.array([landmarks[8].x, landmarks[8].y, landmarks[8].z])
        palm_vector = middle - wrist
        index_vector = index_tip - wrist
        roll = math.degrees(math.atan2(palm_vector[1], palm_vector[0]))
        pitch = math.degrees(math.atan2(palm_vector[2], palm_vector[0]))
        yaw = math.degrees(math.atan2(index_vector[2], index_vector[1]))
        return np.array([roll, pitch, yaw])
    def _smooth_angles(self, new_angles):
        self.prev_angles = (
            self.smoothing * self.prev_angles + (1 - self.smoothing) * new_angles
        )
        return self.prev_angles
    def _adjust_for_hand(self, angles):
        if self.hand_label == "Left":
            angles[0] *= -1
            angles[2] *= -1
        return angles
    def _calibrate(self, current_angles):
        self.zero_angles = current_angles.copy()
        self.calibrated = True
        print("✅ Калібрування виконано. Поточне положення прийнято за нульове.")
    def _apply_scale_and_limits(self, angles):
        scaled = angles * self.scale_factors
        for i, (low, high) in enumerate(self.limits):
            scaled[i] = np.clip(scaled[i], low, high)
        return scaled
    def process_frame(self):
        ret, frame = self.cap.read()
        if not ret:
            return None
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.hands.process(rgb)
        angles_to_return = None
        if results.multi_hand_landmarks:
            landmarks = results.multi_hand_landmarks[0].landmark
            if results.multi_handedness:
                self.hand_label = results.multi_handedness[0].classification[0].label
            angles = self._calculate_angles(landmarks)
            angles = self._adjust_for_hand(angles)
            smooth_angles = self._smooth_angles(angles)
            if not self.calibrated:
                self._calibrate(smooth_angles)
                return frame, None
            relative_angles = smooth_angles - self.zero_angles
            final_angles = self._apply_scale_and_limits(relative_angles)
            self.mp_draw.draw_landmarks(
                frame, results.multi_hand_landmarks[0], self.mp_hands.HAND_CONNECTIONS
            )
            current_time = time.time()
            if (current_time - self.last_time) >= self.delay:
                angles_to_return = final_angles
                self.last_time = current_time
        return frame, angles_to_return
    def release(self):
        self.cap.release()
        cv2.destroyAllWindows()
is_debug = False
controller = HandToPlatformController(delay=0.5)
while True:
    frame, angles = controller.process_frame()
    if frame is None:
        break
    if angles is not None:
        alpha, beta, gamma = angles
        print(f"Кути руки (α, β, γ): {alpha:.1f}, {beta:.1f}, {gamma:.1f}")
        servo_angles = GetAngles(
            is_debug=is_debug,
            is_multi=True,
        ).calculation(alpha=alpha, beta=beta, gamma=gamma)

raise RuntimeError('♡ Код заблоковано у захищеній версії лістингу.')
# --- кінець захищеного фрагмента ---

        if all(not np.isnan(angle) for angle in servo_angles):
            print(
                f"s1: {servo_angles[0]}, s2: {servo_angles[1]}, s3: {servo_angles[2]}"
            )
    cv2.imshow("Hand Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
controller.release()

#
# === rtt_controllers/ai_rtt_old.py
#

import cv2
import mediapipe as mp
import numpy as np
import math
import time
class HandToPlatformController:
```

```
    def __init__(self, camera_index=0, smoothing=0.8, delay=2.5):
        self.cap = cv2.VideoCapture(camera_index)
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(max_num_hands=1)
        self.mp_draw = mp.solutions.drawing_utils
        self.prev_angles = np.zeros(3)
        self.smoothing = smoothing
        self.delay = delay
        self.last_time = 0
    def _calculate_angles(self, landmarks):
        wrist = np.array([landmarks[0].x, landmarks[0].y, landmarks[0].z])
        middle = np.array([landmarks[9].x, landmarks[9].y, landmarks[9].z])
        index_tip = np.array([landmarks[8].x, landmarks[8].y, landmarks[8].z])
        palm_vector = middle - wrist
        index_vector = index_tip - wrist
        roll = math.degrees(math.atan2(palm_vector[1], palm_vector[0]))
        pitch = math.degrees(math.atan2(palm_vector[2], palm_vector[0]))
        yaw = math.degrees(math.atan2(index_vector[2], index_vector[1]))
        return np.array([roll, pitch, yaw])
    def _smooth_angles(self, new_angles):
        self.prev_angles = (
            self.smoothing * self.prev_angles + (1 - self.smoothing) * new_angles
        )
        return self.prev_angles
    def process_frame(self):
        ret, frame = self.cap.read()
        if not ret:
            return None
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = self.hands.process(rgb)
        angles_to_return = None
        if results.multi_hand_landmarks:
            landmarks = results.multi_hand_landmarks[0].landmark
            angles = self._calculate_angles(landmarks)
            smooth_angles = self._smooth_angles(angles)
            self.mp_draw.draw_landmarks(
                frame, results.multi_hand_landmarks[0], self.mp_hands.HAND_CONNECTIONS
            )
            current_time = time.time()
            if (current_time - self.last_time) >= self.delay:
                angles_to_return = smooth_angles
                self.last_time = current_time
        return frame, angles_to_return
    def release(self):
        self.cap.release()
        cv2.destroyAllWindows()
controller = HandToPlatformController()
while True:
    frame, angles = controller.process_frame()
    if frame is None:
        break
    if angles is not None:
        print(f"Кути руки (roll, pitch, yaw): {angles}")
    cv2.imshow("Hand Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break
controller.release()

#
=========================================
=========
# === send_to_arduino/ports_work.py
#
=========================================
=========

import serial.tools.list_ports
def _get_com_ports():
    ports = serial.tools.list_ports.comports()
    return [port.device for port in ports]

#
=========================================
=========
# === send_to_arduino/to_arduino.py
#
=========================================
=========

import serial
import time
from typing import Optional, Dict
class WorkWithArduino:
    def __init__(
        self,
        angles: Optional[tuple[float, float, float]] = None,
        message: Optional[str] = None,
        port: str = "/dev/ttyUSB0",
        baudrate: int = 9600,
        is_debug: bool = False,
        timeout: float = 1.0,
        is_inf: bool = False,
    ):
        self.angles = angles
        self.message = message
        self.port = port
        self.baudrate = baudrate
        self.timeout = timeout
        self.is_debug = is_debug
        self.is_inf = is_inf
        self.ser = None
    def connect(self):
        if self.ser and self.ser.is_open:
            if self.is_debug:
                print(f"[DEBUG] Порт {self.port} уже відкрито.")
            return
        try:
            self.ser = serial.Serial(self.port, self.baudrate, timeout=self.timeout)
            time.sleep(2)
            if self.is_debug:
                print(
                    f"[DEBUG] Підключено до {self.port} (baud={self.baudrate}, timeout={self.timeout})"
                )
        except serial.SerialException as e:
```

```
            print(f"❌ Помилка підключення: {e}")
            self.ser = None
    def send(
        self, angles: Optional[Dict[str, float]] = None,
message: Optional[str] = None
    ):
        """[пояснення вилучено для скороченого
лістингу]"""
        if not self.ser or not self.ser.is_open:
            print("⚠️ Немає з'єднання.")
            return
        angles_to_send = angles if angles is not None else
self.angles
        message_to_send = message if message is not
None else self.message
        if angles_to_send:
            data = self._format_angles(angles_to_send)
        elif message_to_send:
            data = self.message.strip()
            if not data.endswith("|"):
                data += "|"
        else:
            raise ValueError("Нічого не передано для
відправки (ні angles, ні message)")
        self.ser.write(data.encode())
        if self.is_debug:
            print(f"[DEBUG] Надіслано: {data}")
    def receive(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        if self.ser and self.ser.in_waiting > 0:
            data = self.ser.readline().decode().strip()
            print(f"📨 Отримано: {data}")
            return data
        return None
    def close(self):
        """[пояснення вилучено для скороченого
лістингу]"""
        if self.ser and self.ser.is_open:
            self.ser.close()
            if self.is_debug:
                print(f"[DEBUG] Порт {self.port} закрито.")
    def _format_angles(self, angles: Dict[str, float]) ->
str:
        """[пояснення вилучено для скороченого
лістингу]"""
        if not angles:
            raise ValueError("Словник angles не може
бути порожнім")
        formatted = ";".join(str(angles.get(k, 0)) for k in
("s1", "s2", "s3")) + "|"
        if self.is_debug:
            print(f"[DEBUG] Форматовані кути:
{formatted}")
        return formatted
    def main(
        self,
        angles: Dict[str, float] = None,
        message: str = None,
    ):
        self.connect()
        if not self.ser:
            return
        try:
            if self.is_inf:
                if self.is_debug:
                    print("♾️ Безперервний режим роботи")
                while True:
                    self.send(angles=angles,
message=message)
                    self.receive()
                    time.sleep(self.timeout)
            else:
                if self.is_debug:
                    print("🚀 Одноразовий режим")
                self.send(angles=self.angles,
message=self.message)
                self.receive()
        except KeyboardInterrupt:

raise RuntimeError('♡ Код заблоковано у захищеній
версії лістингу.')
# --- кінець захищеного фрагмента ---

            if self.is_debug:
                print("\n⛔ Зупинено користувачем")
        finally:
            self.close()
```