Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Кваліфікаційна наукова
праця на правах рукопису

ВІТЮК АЛЬОНА ЄВГЕНІЇВНА

УДК 004.89

ДИСЕРТАЦІЯ

МЕТОДИ І ПРОГРАМНІ ЗАСОБИ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ РОБОТИЗОВАНОЮ КІНЦІВКОЮ

121 Інженерія програмного забезпечення
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Альона ВІТЮК

Науковий керівник Дорошенко Анатолій Юхимович, д. ф.-м. н., професор

Київ - 2024

АНОТАЦІЯ

Вітюк А.Є. Методи і програмні засоби для автоматизації управління роботизованою кінцівкою. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення». Робота виконана на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» Міністерства освіти і науки України.

Дисертацію присвячено дослідженню програмних засобів для управління роботизованою кінцівкою та методам автоматизації процесу їх розробки з використанням алгоритмів нейроеволюції.

Класичні моделі програмних засобів для управління роботизованими кінцівками базуються на точних специфікаціях завдань і точних моделях цільових об’єктів, з якими взаємодіє робот. Такі підходи гарно працюють у лабораторних умовах, проте мають обмежену придатність в умовах реального світу. Нові програмні застосунки для управління роботизованими кінцівками повинні працювати з недосконалими датчиками та виконавчими механізмами. Розробка моделей управління роботизованою кінцівкою нового покоління має бути автоматизованою, адже завдання та робочі середовища для роботів ускладнюються, зростає потреба в методах навчання та пошуку, які можуть планувати досягнення цілі, не покладаючись на вже існуючу структуру підзадач, розроблену людиною.

Таким чином, вказані задачі визначають актуальну науково-технічну задачу вдосконалення теоретичних (методів) та практичних (програмних засобів) основ автоматизованої розробки адаптивних моделей управління роботизованою системою з захватним пристроєм, яка вирішується у даній дисертаційній роботі.

Метою дисертаційної роботи є підвищення ефективності розроблення програмного управління роботизованою рукою, яка отримує інформацію про

стан середовища з камери, шляхом використання методів автоматизації створення нейроеволюційних моделей.

Результати викладено у чотирьох розділах дисертації.

Перший розділ присвячено огляду та аналізу проблем автоматизованого управління роботизованою рукою робота. Розглянуто особливості взаємодії агента та середовища. Запропоновано методи навчання інтелектуального роботизованого агенту для виконання задач маніпуляції цільовими об’єктами роботизованою рукою. Розглянуто програмні засоби для навчання нейроконтролерів роботизованих систем.

Другий розділ присвячено огляду методів комп’ютерного зору для обробки візуальної інформації, яка може бути використана роботизованим агентом як вхідні дані стану середовища. Розглянуто залежність точності вхідних даних з візуального сенсора та якості позиціонування роботизованої системи. Запропоновано методи та програмні засоби для оцінки впливу похибок параметрів камери на якість реконструйованої моделі середовища.

Третій розділ присвячено методам нейроеволюції для управління роботизованою системою з маніпулятором. Розглянуто алгоритм NEAT для автоматизації розробки нейроконтролерів для управління роботизованою кінцівкою. Розроблено навчальне середовище для автоматизованої розробки моделі позиціонування роботизованої руки. Представлено методи адаптації нейроеволюційного підходу до використання в цільових задачах роботизованої руки.

В четвертому розділі розглядається розроблення програмного засобу для адаптивного навчання контролерів роборуки на основі нейромереж з використанням методів нейроеволюціїї для задач, де вхідні дані представлені у вигляді зображень з камери. Представлена реалізація програмного рішення для навчання роботизованої руки заданої конфігурації виконання задач у тренувальних середовищах.

У дисертаційній роботі отримано низку **нових наукових результатів**, а саме:

Вперше запропоновано метод пошуку новизни в нейроеволюції для автоматизації створення моделі програмного управління роботизованою кінцівкою, що дозволяє прискорити процес розробки системи управління для нових задач роботизованої кінцівки, зокрема для адаптації конфігурації кінцівки або середовища, що підтверджується за допомогою навчання у тестових середовищах та оцінки отриманої моделі. Представлений метод відрізняється від існуючих тим, що не потребує ресурсів розробника для ручного налаштування параметрів управління роботизованою кінцівкою з метою отримання найбільш ефективної стратегії вирішення кожної окремої задачі управління.

Вперше запропоновано метод навчання на основі гіперкуба для програмного управління в задачах агента-роборуки, що отримує інформацію про стан середовища з камери. Використання методу на основі гіперкуба для програмного управління агентом на основі зображень з камери покращує ефективність навчання моделі за рахунок використання відображення геометрії фенотипу субстрату штучної нейронної мережі на його шаблон зв'язків на основі гіперкубу, сприяючи розвитку більш універсальних і потужних архітектур нейронних мереж.

Вперше розроблено метод підвищення якості вхідних даних з камери, які використовуються для автоматизованого управління роботизованою кінцівкою, що надає можливість навчання роботизованих агентів на тестових середовищах в умовах, що наближені до реальних, адже враховують похибки камери. Цей метод дозволяє налаштовувати конфігурацію камери у навчальній системі для адаптації моделі до реальної конфігурації роботизованої системи, що забезпечує зменшення похибок параметрів камери під час калібрування та суттєве підвищення успішності стійкого захвату кінцівкою.

Практичне значення одержаних результатів полягає у експериментально підтвердженій ефективності представлених методів на відомих тестових середовищах OpenAI Gym для перевірки якості нейроеволюційних алгоритмів як для двовимірних моделей середовищ, так і

для тривимірних, де інформацію про стан середовища роботизований агент отримує з камери, що наближено до роботи системи в умовах реального світу. Отримано патент на засоби калібрування камери для підвищення якості вхідних даних з камери, що використовується для автоматизованого управління роботизованою кінцівкою. Реалізовано метод пошуку новизни під час автоматизованого навчання моделі системи для двовимірного середовища та метод на основі гіперкуба для тривимірного середовища при виконанні маніпуляційних задач роботизованою кінцівкою. Встановлено, що використання представлених методів у нейроеволюційному процесі для вирішення задачі позиціонування дозволяє підвищити ефективність процесу навчання мережі та отримати оптимальну топологію управляючої мережі.

Основні результати дисертаційної роботи опубліковано у 9 наукових працях, зокрема, у 5 наукових статтях, з яких 4 статті опубліковано у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б», та 1 статтю у інших виданнях. Крім того, 3 статті у матеріалах науково-технічних конференцій. Також опубліковано 1 патент.

Ключові слова: автоматизація розробки моделі, нейромережеві моделі управління, роботизована кінцівка, калібрування камери, нейроеволюція, NEAT, комп'ютерний зір, алгоритм навчання.

SUMMARY

Vitiuk A. Methods and software tools for automating control of a robotic arm. Qualifying scientific work on the rights of the manuscript.

Ph.D. thesis in the field of knowledge 12 Information technologies in a specialty 121 Software engineering. – National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, 2023.

The dissertation is devoted to the research of software tools for controlling a robotic limb and methods of automating the process of their development using neuroevolution algorithms.

Classical software models for controlling robotic limbs are based on accurate task specifications and accurate models of the target objects with which the robot interacts. Such approaches work well in laboratory settings, but have limited applicability in real-world settings. New robotic applications require controllers that can work with imperfect sensors and actuators in changing environments. As tasks and work environments for robots become more complex, there is a growing need for learning and retrieval methods that can plan to achieve a goal without relying on a pre-existing structure of human-designed subtasks.

Thus, the specified tasks determine the actual scientific and technical task of improving the theoretical (methods) and practical (software) foundations of the automated development of adaptive control models of a robotic system with a gripping device, which is solved in this dissertation work.

The purpose of the dissertation is to increase the effectiveness of the development of a neural network model in the means of software control of a robotic arm by using methods of automating the creation of neuroevolutionary control models for a robotic agent that receives information about the state of the environment from a camera.

The results are presented in four sections of the thesis.

The first section is devoted to the review and analysis of the problems of automated control of a robotic arm of a robot. The peculiarities of the interaction between the agent and the environment are considered. Methods of training an

intelligent robotic agent to perform the tasks of manipulation of target objects with a robotic arm are proposed. Software tools for training neurocontrollers of robotic systems are considered.

The second section is devoted to an overview of computer vision techniques for processing visual information that can be used by a robotic agent as input to the state of the environment. The dependence of the accuracy of the input data from the visual sensor and the positioning quality of the robotic system is considered. Methods and software tools are proposed for evaluating the impact of camera parameter errors on the quality of the reconstructed environment model.

The third section is devoted to neuroevolutionary methods for controlling a robotic system with a manipulator. The NEAT algorithm for automating the development of neurocontrollers for controlling a robotic limb is considered. A learning environment for the automated development of a robotic arm positioning model has been developed. The methods of adapting the neuroevolutionary approach to the use of a robotic arm in target tasks are presented.

The fourth section deals with the development of a software tool for adaptive training of robot controllers based on neural networks using neuroevolution methods for tasks where the input data is presented in the form of camera images. The implementation of a software solution for training a robotic hand in a given configuration of performing tasks in training environments is presented.

A number of **new scientific results** were obtained in the dissertation work, namely:

For the first time, a method of searching for novelty in neuroevolution is proposed to automate the creation of a model of software control of a robotic limb, which allows to accelerate the process of developing a control system for new tasks of a robotic limb, in particular for adapting the configuration of the limb or the environment, which is confirmed by means of training in test environments and evaluation of the obtained model. The presented method differs from the existing ones in that it does not require developer resources to manually adjust the control

parameters of the robotic limb in order to obtain the most effective strategy for solving each individual control problem.

For the first time, a learning method based on a hypercube is proposed for software control in the tasks of a robotic agent that receives information about the state of the environment from a camera. The use of a hypercube-based method for programmatic control of an agent based on camera images improves model training efficiency by exploiting the mapping of the phenotype geometry of an artificial neural network substrate to its hypercube-based connection template, contributing to the development of more versatile and powerful neural network architectures.

For the first time, a method for improving the quality of input data from a camera used for automated control of a robotic limb has been developed, which provides the opportunity to train robotic agents on test environments in conditions close to real ones, because camera errors are taken into account. This method allows you to adjust the camera configuration in the training system to adapt the model to the real configuration of the robotic system, which ensures a reduction in camera parameter errors during calibration and a significant increase in the success rate of stable limb grasping.

The practical significance of the obtained results lies in the experimentally confirmed effectiveness of the presented methods on the well-known OpenAI Gym test environments for checking the quality of neuroevolutionary algorithms both for two-dimensional models of environments and for three-dimensional ones, where the robotic agent receives information about the state of the environment from the camera, which approximates the operation of the system in real world conditions. Received a patent for camera calibration tools to improve the quality of input data from a camera used for automated control of a robotic limb. The method of searching for novelty during automated learning of a system model for a two-dimensional environment and a method based on a hypercube for a three-dimensional environment when performing manipulation tasks with a robotic limb are implemented. It was established that the use of the presented methods in the neuroevolutionary process to

solve the positioning problem allows to increase the efficiency of the network learning process and obtain the optimal topology of the control network.

The **main results of the dissertation work were published** in 9 scientific works, in particular, in 5 scientific articles, of which 4 articles were published in specialized publications included in the list of scientific specialized publications of Ukraine with the assignment of category "B", and 1 article in other publications. In addition, 3 articles in the materials of scientific and technical conferences. Also published 1 patent.

Keywords: automation of model development, neural network control models, robotic arm, camera calibration, neuroevolution, NEAT, computer vision, learning algorithm.

Список публікацій здобувача / List of publications of the applicant:

Статті у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б» / the articles published in scientific journals included in the list of scientific journals of Ukraine in category «Б»):

1. Вітюк, А. Є., Корнага, Я. І., & Барабаш, А. О. (2018). Захоплення невідомих об'єктів мобільним роботом із використанням візуальної інформації. Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, (29 (68),№ 1 (1)), 93-98 (Фахове видання, «Б»).
2. Вітюк, А. Є., & Дорошенко, А. Ю. (2022). Програмний пакет для оцінки похибки калібрування стереокамери в системі комп'ютерного зору. Проблеми програмування (3-4), 469-477 (Фахове видання, «Б»).
3. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Пошук новизни у методі нейроеволюції для позиціонування роботизованої кінцівки. Проблеми програмування (3), 49-57 (Фахове видання, «Б»).
4. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Програмний пакет для адаптивного навчання контролерів роборуки на основі нейромереж. Проблеми програмування (4), 98-107 (Фахове видання, «Б»).

Публікація у матеріалах наукової конференції / the publication in the proceedings of the scientific conference:

5. Корнага Я.І., Вітюк А.Є. (2018). Оцінка якості стійкого захвату, спланованого на основі тривимірної реконструкції цільового об'єкту по зображенням з монокамери, Міжнародна науково-технічна конференція «The International Conference on Security, Fault Tolerance, Intelligence".
6. Alona Vitiuk, Anatoliy Doroshenko. (2022) Software Package for Evaluation the Stereo Camera Calibration for 3D Reconstruction in Robotics Grasping System. CEUR Workshop Proceedings Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG.

7. Вітюк А.Є., Дорошенко А.Ю. (2023). Використання нейроеволюції при пошуку політик в формі нейромереж для управління робочою кінцівкою. Перша Науково-теоретична конференція «Пріоритети і виклики реалізації Стратегії розвитку штучного інтелекту в Україні». Artificial Intelligence (2).

Праці, які додатково відображають результати дисертації / works that additionally reflect the results of the dissertation:

8. US10841570B2, Petro Kytsun Iegor Vdovychenko, Alona Vitiuk, Nataliya Sakhnenko, Oleksii Panfilov, «Simultaneous camera and sensors calibration apparatus and method», 2018
9. Вітюк А.Є., Корнага Я.І. (2018). Оцінка впливу похибок калібрування на тривимірну реконструкцію у монокулярній системі одночасної локалізації та картографування. Східноєвропейський науковий журнал (4 (32)), 29–35.

ЗМІСТ

ЗМІСТ 12
ВСТУП 14
РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ АВТОМАТИЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ РОБОТИЗОВАНИМ АГЕНТОМ 21
1.1 Основні методи розробки програмного забезпечення для управління роботизованими агентами 21
1.2 Основні методи еволюційного глибокого навчання для розробки програмних засобів автоматизованого управління агентом 28
1.3 Основні стратегії застосування еволюційного глибокого навчання 39
1.4 Висновки до першого розділу 41
РОЗДІЛ 2. МЕТОДИ КОМП'ЮТЕРНОГО ЗОРУ ТА ПРОГРАМНІ ЗАСОБИ ДЛЯ УПРАВЛІННЯ РОБОТИЗОВАНОЮ КІНЦІВКОЮ З ВИКОРИСТАННЯМ ДАНИХ З КАМЕРИ 44
2.1 Основні методи використання зображень для задачі позиціонування мобільного роботизованого агента 44
2.2 Основні методи програмного управління роботизоваим агентом з камерою у середовищі 47
2.3 Методи калібрування камери для зменшення впливу похибок даних із зображень 52
2.4 Оцінка впливу якості калібрування камери на точність системи з комп'ютерним зором 57
2.5 Оцінювання успішності захвату цільового об'єкту, здійсненого на основі зображень з камери 65
2.6 Висновки до другого розділу 67
РОЗДІЛ 3. МЕТОДИ АДАПТИВНОГО НАВЧАННЯ ТА ПРОГРАМНІ ЗАСОБИ УПРАВЛІННЯ РОБОРУКОЮ НА ОСНОВІ НЕЙРОЕВОЛЮЦІЇ .. 69
3.1 Нейроеволюційний підхід для автоматизації розробки нейроконтролерів 69

3.2 Програмні засоби для навчання нейроконтролерів роботизованих систем 72
3.3 Опис навчального середовища для задачі позиціонування роботизованої руки 75
3.4 Адаптивний підхід з використанням алгоритму NEAT 78
3.5 Застосування методу оптимізації пошуком новизни для алгоритму NEAT 82
3.6 Програмний засіб для адаптивного навчання нейромережевого засобу управління агентом-роборукою 84
3.7 Оцінювання ефективності пошуку новизни для двовимірної задачі досягнення цільового положення агентом-роборукою 90
3.7 Висновки до третього розділу 97
РОЗДІЛ 4. МЕТОД АДАПТИВНОГО НАВЧАННЯ МОДЕЛІ У ПРОГРАМНИХ ЗАСОБАХ УПРАВЛІННЯ РОБОРУКОЮ З ВИКОРИСТАННЯМ ДАНИХ З КАМЕРИ 100
4.1 Метод HyperNEAT для системи з вхідними даними у форматі зображень 100
4.2 Застосування методу на основі гіперкуба для адаптивного навчання роботизованого агенту 104
4.3 Програмний засіб для адаптивного навчання агента-роборуки на основі вхідних зображень 109
4.4 Оцінювання ефективності методу навчання на основі гіперкуба для програмного управління в задачах роборуки на основі даних з камери 114
4.5 Висновки до четвертого розділу 122
ВИСНОВКИ 124
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 127
ДОДАТКИ 138

ВСТУП

Актуальність теми. Дані сьогодні відіграють важливу роль і необхідні для створення та оцінки програмного забезпечення та моделей. Причому, розміри даних поступово збільшуються, щоб моделі розроблених систем краще відображали умови реального світу. Таке збільшення даних створює додаткові складнощі для розробників. Глибокі нейронні мережі, які можуть вивчати багатовимірні представлення, працюють набагато краще, ніж інші алгоритми, у багатьох сферах, зокрема таких як комп'ютерне бачення, що використовується для обробки даних з камери роботизованої системи. Ці моделі надто великі та складні із сотнями мільйонів параметрів, які потребують великої кількості обчислювальних ресурсів. Крім того, їх продуктивність сильно залежить від архітектури мережі та конфігурації параметрів. Отже, пошук правильної топології та гіперпараметрів по суті зводиться до процесу оптимізації чорного ящика.

Ці параметри конфігурації зазвичай визначаються з попередніх досліджень, оскільки ручне тестування та оцінка є виснажливим і трудомістким процесом, який потребує досвіду та знань. Альтернативний підхід полягає у використанні алгоритмів нейроеволюції для допомоги глибоким нейронним мережам у побудові архітектури та вивчення гіперпараметрів.

Такий підхід може бути використаний для актуальних задач робототехніки, таких як планування стійкого захвату та ефективне переміщення об'єктів в неструктурованому середовищі, адже рівень точності та ефективності роботизованої руки все ще не може досягнути рівня людської руки. А з огляду на активну автоматизацію промисловості, попит на роботизовані системи, які можуть виконувати складні та точні завдання, посилився. Від виробництва до охорони здоров'я, від логістики до аерокосмічної галузі, розгортання роботизованих систем для задач маніпулювання об'єктами стало повсюдним, революціонізуючи традиційні робочі процеси та відкриваючи нові можливості.

Більшість роботизованих завдань вимагають виконання багатокрокових цілей, для яких потрібен механізм планування. Традиційні роботизовані контролери зазвичай розробляються вручну, а розподіл завдань на підзадачі виконує людина-конструктор. Оскільки завдання та робочі середовища для роботів ускладнюються, зростає потреба в методах навчання та пошуку, які можуть планувати вирішення задачі, не покладаючись на вже існуючу структуру підзадач або станів, розроблену людиною.

Складність, притаманна сучасним роботам, вимагає управляючих програмних засобів, які виходять за межі традиційних методологій, що засновані на правилах. Інтелектуальні моделі управління, засновані на передових технологіях, таких як штучний інтелект, машинне навчання та нейронні мережі, обіцяють подолати розрив між можливостями роботизованої руки та вимогами складних сценаріїв реального світу [1].

Задача позиціонування роборуки та маніпулювання цільовим об’єктом характеризується своєю складністю, передбачаючи точний контроль кількох ступенів свободи та взаємодії з динамічним середовищем невідомої конфігурації. Нейронні мережі, з їхньою здатністю навчатися та представляти складні залежності від вхідних даних, пропонують автоматизований шлях для вирішення цих проблем. Крім того, адаптивність структур нейронної мережі додатково надає таким системам здатність динамічно налаштовувати свою архітектуру відповідно до вимог різноманітних завдань, що можуть змінюватися.

Важливість автоматизованої розробки підкреслюється потребою в управляючих програмних засобах, які можуть легко адаптуватися до мінливого середовища, різних форм об’єктів маніпулювання і факторів, пов’язаних із середовищем. Навчальні можливості нейронних мереж, особливо з адаптивними структурами, роблять їх добре придатними для сценаріїв, де класичне програмування не може забезпечити знаходження мінімальної моделі, розмір якої є критично важливим для систем реального часу [2].

На відміну від класичних підходів, у даній роботі розглядається реалізація системи навчання з використанням нейроеволюції при створенні контролера для управління роботизованою рукою. Використання еволюційних алгоритмів є особливо ефективним для задач з використанням роботизованої руки, у яких кінцеве положення може бути досягнуто багатьма оптимальними способами, а отже потребує навчання з підкріпленням. Крім того нейроеволюційний підхід дозволяє отримати оптимальну топологію мережі, яка робить модель більш ресурсоефективною та легшою для аналізу [3]. Отже дослідження розробки автоматизованих засобів управління роботизованою кінцівкою представляє особливий інтерес.

Зв'язок роботи з науковими програмами, планами, темами. Результати дослідження за темою дисертаційної роботи впроваджено у навчальний процес кафедри інформаційних систем та технологій факультету інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Крім того, результати, отримані у дисертаційному дослідженні, реалізовані при впровадженні розробки автоматизованого управління роботизованими системами згідно актів (Додаток А).

Мета і задачі дослідження. Метою дисертаційної роботи є підвищення ефективності розроблення програмного управління роботизованою рукою, яка отримує інформацію про стан середовища з камери, шляхом використання методів автоматизації створення нейроеволюційних моделей.

Відповідно до поставленої мети основними задачами дослідження є:

- розробити метод автоматизації створення нейромереж для програмного управління роботизованою кінцівкою.
- розробити метод підвищення точності системи автоматизованої розробки нейромережевого програмного управління роботизованою кінцівкою з використанням вхідних даних з камери.

- розробити програмну реалізацію для оцінки впливу похибок параметрів камери на точність програмного управління роботизованою кінцівкою.
- розробити програмну реалізацію для автоматизованого навчання нейромережевої моделі системи для виконання маніпуляційних задач роботизованою кінцівкою у навчальних середовищах.
- виконати експериментальні роботи з перевірки ефективності розроблених методів на відомих тестових середовищах OpenAI Gym.

Предметом дослідження є використання нейроеволюційного підходу та методів комп'ютерного зору для автоматизації розробки програмного управління роботизованою кінцівкою робота, оснащеного камерою.

Об'єктом дослідження є методи та програмні засоби для автоматизації розробки програмного управління та підвищення якості вхідних даних у форматі зображень для роботизованих систем з робочою кінцівкою-маніпулятором та камерою з використанням методів нейроеволюції та комп'ютерного зору для двовимірних та тривимірних середовищ.

Методи дослідження: теорія програмування, теорія алгоритмів, теорія нейронних мереж, машинне навчання.

Наукова новизна полягає в наступному:

1. **Вперше** запропоновано метод пошуку новизни в нейроеволюції для автоматизації створення моделі програмного управління роботизованою кінцівкою, що дозволяє прискорити процес розробки системи управління для нових задач роботизованої кінцівки, зокрема для адаптації конфігурації кінцівки або середовища, що підтверджується за допомогою навчання у тестових середовищах та оцінки отриманої моделі. Представлений метод відрізняється від існуючих тим, що не потребує ресурсів розробника для ручного налаштування параметрів управління роботизованою кінцівкою з

метою отримання найбільш ефективної стратегії вирішення кожної окремої задачі управління.

2. **Вперше** запропоновано метод навчання на основі гіперкуба для програмного управління в задачах агента-роборуки, що отримує інформацію про стан середовища з камери. Використання методу на основі гіперкуба для програмного управління агентом на основі зображень з камери покращує ефективність навчання моделі за рахунок використання відображення геометрії фенотипу субстрату штучної нейронної мережі на його шаблон зв'язків на основі гіперкубу, сприяючи розвитку більш універсальних і потужних архітектур нейронних мереж.
3. **Вперше** вперше розроблено метод підвищення якості вхідних даних з камери, які використовуються для автоматизованого управління роботизованою кінцівкою, що надає можливість навчання роботизованих агентів на тестових середовищах в умовах, що наближені до реальних, адже враховують похибки камери. Цей метод дозволяє налаштовувати конфігурацію камери у навчальній системі для адаптації моделі до реальної конфігурації роботизованої системи, що забезпечує зменшення похибок параметрів камери під час калібрування та суттєве підвищення успішності стійкого захвату кінцівкою.

Практичне значення одержаних результатів полягає у експериментально підтвердженій ефективності представлених методів на відомих тестових середовищах OpenAI Gym для перевірки якості нейроеволюційних алгоритмів як для двовимірних моделей середовищ, так і для тривимірних, де інформацію про стан середовища роботизований агент отримує з камери, що наближено до роботи системи в умовах реального світу. Отримано патент на засоби калібрування камери для підвищення якості вхідних даних з камери, що використовується для автоматизованого

управління роботизованою кінцівкою. Реалізовано метод пошуку новизни під час автоматизованого навчання моделі системи для двовимірного середовища та метод на основі гіперкуба для тривимірного середовища при виконанні маніпуляційних задач роботизованою кінцівкою. Встановлено, що використання представлених методів у нейроеволюційному процесі для вирішення задачі позиціонування дозволяє підвищити ефективність процесу навчання мережі та отримати оптимальну топологію управляючої мережі.

Особистий внесок здобувача. Всі основні результати дисертаційного дослідження, які представлені до захисту, одержані автором особисто. У публікаціях, написаних у співавторстві, здобувачеві належать наступні результати. У роботі [1] здобувачем запропоновано використання методу новизни для алгоритму NEAT, що використовується для навчання управляючої нейромережі двовимірною роборукою. У роботі [2] здобувачем запропоновано метод та програмний засіб для використання алгоритму HyperNEAT для навчання тривимірної системи з роборукою, яка отримує інформацію про стан середовища з камери. У роботі [4] здобувачем запропоновано методи на основі нейроеволюції для навчання моделі у програмних засобах управління роборукою. У роботі [9] здобувачем запропоновано алгоритмічне забезпечення системи управління для здійснення стійкого захвату роборукою з вхідною інформацією у вигляді зображень з камери. У роботі [24] здобувачем запропоновано метод та програмну реалізацію для оцінки впливу похибок калібрування камери на якість інформації, що може бути отримана із зображення камери. У роботі [56] здобувачем запропоновано алгоритмічне забезпечення оцінки якості калібрування камери для стереозображень. У роботі [57] здобувачем запропоновано методи оцінки якості захвату цільових об'єктів роборукою на основі зображень з монокамери. У роботі [99] здобувачем реалізовано методику калібрування пристрою з багатьма камерами. У роботі [98] здобувачем запропоновано метод оцінки точності моделі об’єкта для задачі стійкого захвату у комбінованій системі пропозиції захвату та реконструкції тривимірної моделі об’єкта.

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідалися та обговорювалися на міжнародних та національних науково-практичних конференціях:

1. Міжнародна науково-технічна конференція «The International Conference on Security, Fault Tolerance, Intelligence», Київ, 2018.
2. CEUR Workshop Proceedings Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG, Київ, 2022.
3. Перша науково-теоретична конференція «Пріоритети і виклики реалізації стратегії розвитку штучного інтелекту в Україні», Artificial Intelligence, Київ, 2023

Публікації. Основні результати дисертаційної роботи опубліковано у 9 наукових працях, зокрема, у 5 наукових статтях, з яких 4 статті опубліковано у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б», та 1 статтю у інших виданнях. Крім того, 3 статті у матеріалах науково-технічних конференцій. Також опубліковано 1 патент.

Структура роботи.. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, 6 додатків та списку використаних джерел, що включає 99 найменувань. Загальний обсяг роботи становить 185 сторінок, основний текст роботи викладено на 138 сторінках. Робота містить 27 рисунків та 6 таблиць.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ АВТОМАТИЗАЦІЇ РОЗРОБКИ ПРОГРАМНИХ МОДЕЛЕЙ УПРАВЛІННЯ РОБОТИЗОВАНИМ АГЕНТОМ

1.1 Основні методи розробки програмних моделей управління роботизованими агентами

В останні роки машинне навчання зробило революцію у сфері робототехніки та автоматизації. Використовуючи алгоритми, роботів можна навчити виконувати різні завдання і навіть навчатися самостійно. Це дозволило створити більш досконалих роботів, які можуть самостійно орієнтуватися в складних середовищах, взаємодіяти з людьми більш природним чином, а також виконувати виробничі завдання більш ефективно.

Машинне навчання дозволяє роботам обробляти величезні обсяги даних у реальному часі, дозволяючи їм приймати швидші та точніші рішення. Це допомагає роботам краще розуміти навколишнє середовище та об’єкти навколо них. Наприклад, їх можна запрограмувати ідентифікувати об’єкти за допомогою комбінації візуальних, тактильних і звукових датчиків. Це дозволяє розпізнавати різні об’єкти в навколишньому середовищі та відповідно реагувати [4].

Машинне навчання також дозволяє роботам вчитися на їхньому досвіді. Використовуючи дані, зібрані з попередніх завдань, роботи можуть коригувати свою поведінку, щоб краще виконувати майбутні завдання. Це особливо корисно в областях, де роботам потрібно швидко адаптуватися до мінливих умов і середовища. Більшість робототехнічних завдань вимагають багатокрокових цілей, які вимагають механізму планування. Традиційні роботизовані контролери зазвичай проектуються вручну, а розподіл завдань на підзадачі виконує людина [46].

Багато сценаріїв вимагають адаптації роботів до нових умов або навіть навчання абсолютно новій поведінці. Роботу, який виробляє автомобілі, наприклад, час від часу доведеться адаптуватися до нових моделей автомобілів.

Для багатьох реальних додатків достатньо запрограмувати необхідну поведінку вручну, але часто це неможливо, оскільки середовище може просто змінюватися надто часто або навіть бути заздалегідь невідомим інженерам, які програмують систему

1.1.1 Основні задачі роботизованого агента з робочою кінцівкою для взаємодії з цільовими об’єктами

Сучасні роботи можуть бути запрограмовані для виконання багатьох складних завдань по маніпулюванню, починаючи від використання інструментів для збирання складних машин, і до проведення високоточних медичних операцій [8]. Але, автономне захоплення передчасно невідомого об’єкта з використанням мінімального набору датчиків, що отримують інформацію про стан навколишнього середовища, все ще залишається складною проблемою.

При спробі захоплення об’єкту, характеристики якого є завчасно відомими, або якщо відома повна тривимірна модель об’єкта, то існують алгоритми з різними підходами проведення процедури захоплення. Але, на практиці отримання повної та точної реконструкції об’єкта, що є невідомим для робота, є окремою складною задачею. Для пасивних стереосистем, тривимірна реконструкція об’єктів з малотекстурованою поверхнею є нерозв’язною задачею. У варіанті, коли стереопсис дає задовільний результат, реконструюються зазвичай лише видимі ділянки об’єкта.

Автоматичне захоплення невідомого об’єкту за допомогою одного зображення є складною проблемою, тому що положення та форма об’єкта є невідомими та можливі конфігурації захоплюючого пристрою, для здійснення жорсткого захоплення, можуть варіюватися. При цьому для рухомого робота, що переміщується у просторі, постановка задачі розрахунку точок захоплення має відмінні умови. На вхід такої системи подається не лише одне зображення сцени з цільовим об’єктом, а послідовність кадрів (відео), за допомогою якого

можливо отримати зображення об’єкту з різних ракурсів та розрахувати щільну тривимірну реконструкцію [9].

В останні десятиліття проблема автоматичного захоплення невідомих об’єктів стала відігравати все більшу роль в машинному баченні. Більшість розглянутих алгоритмів використовують певний вид навчання для здійснення стійкого захоплення. Оскільки захоплення передбачає контакт та сили пальців на поверхню об’єкта, то важливо мати достовірну інформацію як про руку, так і про поверхню цільового об’єкта [9]. Необхідно враховувати обмеження руки робота та конфігурації об’єкта, а також обмеження поставленого завдання [10]. Незважаючи на наявність підходу зі здійснення захоплення лише по двовимірним зображенням [11, 12], більшість технік полягають у обробці тривимірних даних. З огляду на складність поставленої задачі, багато робіт були спрямовані на спрощення тривимірної форми, зокрема плоских представленнях [13] та компланарності, що комбінується з інформацією про колір [14]. Інші підходи передбачають наявність точної моделі об’єкта або такої, що складається з високорівневих примітивів. Прикладом є використання принципу синергії руки [15, 16, 17]. Також існує підхід із застосуванням машинного навчання, зокрема алгоритму Support Vector Machines (SVM) та використанням примітивів форм високого рівня [18]. Алгоритм з навчанням на основі якості двовимірних захоплень, що базується на нейронних мережах та генетичному алгоритму, представлений у роботі «Real-Time Visual Grasp Synthesis Using Genetic Algorithms and Neural Networks» [19].

Задача побудови реконструкції навколишнього середовища роботом є актуальною та активно досліджується. Оскільки камери глибини не є настільки поширеними, як кольорові камери, багато робіт пов’язані зі щільними та напівщільними методами одночасної локалізації та картографування (simultaneous localization and mapping - SLAM) по зображенням з однієї камери [20, 21, 22].

1.1.2 Основні принципи програмного представлення роботизованого агента

Розглянемо модель роботизованої руки, представлену як інтелектуальний агент в обмеженому середовищі. Агент — це суб’єкт, який здатний сприймати своє оточення за допомогою датчиків, а також впливати на своє оточення за допомогою приводів. Агенти зазвичай можуть сприймати власні дії, але не обов'язково вплив цих дій на навколишнє середовище. Агент може бути математично описаний функцією агента, яка визначає дію агента на основі всієї його послідовності сприйняття. Таким чином, поведінка агента може бути повністю описана шляхом визначення дії для кожної можливої послідовності сприйняття. Схема взаємодії агента і середовища представлена на рисунку 1 [4].

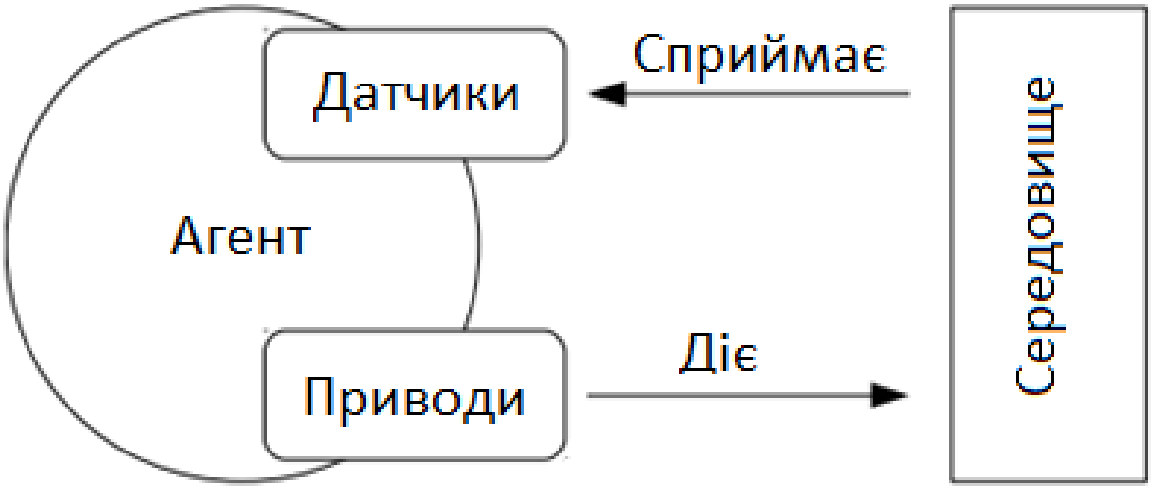


Рис. 1. Взаємодія компонентів навчального середовища

Конкретна реалізація функції агента називається програмою агента. Мета розробки агента полягає в тому, щоб отримати його правильну поведінку. Щоб визначити які функції агента будуть хорошими, а які поганими, зазвичай застосовуються показники ефективності. Вони залежать від типу завдання, яке має виконати агент. Якщо розробляти агента, здатного керувати автомобілем, наприклад, можна базувати показники продуктивності на споживанні палива, часу поїздки і ступеню виконання ПДР. Загалом, поведінка агенту вважається правильною, якщо він успішно виконує поставлене перед ним завдання.

Середовища, які агенти сприймають і з якими взаємодіють, можуть бути дуже різноманітним. Вони можуть бути простими, як настільна гра, або складними, як жвава дорога з великою кількістю автомобілів, пішоходів, дорожніх знаків і ліхтарів. Щоб краще зрозуміти сутність середовища, їх можна класифікувати за кількома властивостями.

Якщо агент може сприймати весь стан навколишнього середовища в будь-який момент часу, він називається повністю спостережуваним, інакше він називається частково спостережуваним. Середовище може бути недоступним для повного спостереження, оскільки датчики агента не здатні вимірювати всі відповідні вхідні дані або через те, що показання датчиків є шумними. Часто також буває так, що показники навколишнього середовища приховані, і тому їх неможливо спостерігати. Повністю спостережуване середовище спрощує завдання агента, оскільки йому не потрібно досліджувати та відстежувати будь-які факти, які раніше зустрічалися.

Середовище називається детермінованим, якщо його стан залежить лише від попереднього стану та дій агента. Інакше середовище називається стохастичним. Водіння автомобіля є прикладом стохастичного середовища, тому що неполадки двигуна можуть відбуватися без передчасних попереджень, а шини інколи можуть лопнути. Якщо середовище є детермінованим та виключає дії інших агентів, воно називається стратегічним середовищем.

Якщо середовище поділяється на незалежні завдання, що складаються з набору входів і однієї дії, воно називається епізодичним. В епізодичних середовищах дії агента ґрунтуються лише на вхідних даних поточного епізоду, але не на попередніх епізодах. Прикладом для епізодичного середовища є агент класифікації, завдання якого полягає в тому, щоб визначити, чи є деталі на конвеєрі якісними чи ні. Протилежністю епізодичності є послідовність. Водіння автомобіля є типовим прикладом для послідовного середовища.

Середовища можуть бути статичними або динамічними. Стан статичного середовища не змінюється допоки агент не діє на нього. Це надає перевагу, адже агенту не потрібно стежити за навколишнім середовищем під час

простою, і прийняття рішення може тривати достатньо довго. Однак більшість середовищ є динамічними, що означає, що поки агент вирішує, що робити далі, відбуваються зміни стану середовища, що може мати негативний вплив на продуктивність агента. Водіння автомобіля також є прикладом динамічного середовища. Якщо автомобіль перед автомобілем агента загальмує, автомобіль агента розіб'ється, якщо він не вирішить вжити відповідних дій вчасно.

Якщо середовище має лише скінченну кількість станів, воно називається дискретним середовищем, інакше воно називається безперервним середовищем. Прикладом дискретного середовища є проста настільна гра, стан якої складається виключно з позицій ігрових фігур на дошці. Водіння автомобіля, з іншого боку, явно не є дискретним середовищем, оскільки можливі позиції та швидкості інших автомобілів на дорозі нескінченні. Взагалі, будь-яка форма змодельованого середовища є дискретною, оскільки комп'ютери не можуть представляти безперервні значення. Однак, оскільки кількість можливих станів у змодельованому 3D-середовищі практично нескінченна, ці середовища також класифікуються як безперервні.

Якщо агент діє на середовище разом з іншими агентами, це називається мультиагентним середовищем. Якщо агент лише один, це називається одноагентним середовищем. Однак іноді незрозуміло, чи слід розглядати інші сутності як агентів чи просто як об'єкти зі стохастичною поведінкою. Якщо агент керує автомобілем по жвавій дорозі, інші транспортні засоби є агентами чи ні? Відповідь на це запитання залежить від того, чи впливають інші суб'єкти на показники ефективності агента чи ні. У прикладі водіння це зазвичай не так, оскільки продуктивність агента (вчасне прибуття, дотримання правил дорожнього руху, оптимізація споживання палива) загалом не залежить від поведінки інших транспортних засобів. Однак якщо вимірювання продуктивності розширено, щоб включати «уникнення зіткнень», це залежить від поведінки інших водіїв, які також є агентами. Середовище, де ключовий аспект показників продуктивності (уникнення зіткнень) призводить до покращення продуктивності для всіх агентів, називається кооперативним

мультиагентним середовищем. Однак у автомобільних перегонах все навпаки, тому що ефективність (перемога в перегонах) одного агента зростає, а продуктивність інших агентів знижується. Таке середовище називається конкурентним мультиагентним середовищем [5].

1.1.3 Огляд підходів до навчання інтелектуального роботизованого агента

Потужні інтелектуальні агенти потужними повинні мати можливість адаптуватися до змін навколишнього середовища. Для цього важливо використовувати відповідні алгоритми навчання. Ці алгоритми використовуються не лише для того, щоб дозволити роботам адаптуватися до навколишнього середовища, але й для вивчення абсолютно нових моделей поведінки, які можуть бути надто складними, щоб задаватися вручну. Як і люди, штучні агенти можуть навчатися різними способами.

Навчання з учителем. У цьому типі алгоритмів навчання, який також відомий як кероване навчання, агент супроводжується навчаючим екземпляром [6]. Цей екземпляр має вбудовані знання про бажану функціональність. Процес навчання ітераційний. У кожній ітерації агенту та вчителю надається приклад вхідного вектора. Потім агент обчислює результат відповідно до своїх поточних знань, а вчитель створює бажаний результат. Різниця між виходом агента та бажаним виходом, тобто значення похибки, потім передається агенту. Використовуючи це значення, агент може поступово покращувати свою поведінку. Зрештою, агент вчиться наслідувати вчителя, який йому більше не буде потрібен. Набір прикладів вхідних значень разом із бажаними вихідними значеннями називається тренувальним набором процесу навчання [7].

Навчання без учителя. У багатьох випадках навчання з учителем неможливе, оскільки немає попередніх знань про цільову функцію. Якщо це так, можна використовувати окремий клас алгоритмів, який відомий як навчання без вчителя [6]. Цю групу алгоритмів можна розділити на навчання з підкріпленням і некероване навчання.

Алгоритмічний клас навчання з підкріпленням є найбільш загальним з усіх алгоритмів навчання і може бути застосований до багатьох проблем реального світу. Навчання з підкріпленням базується на функції витрат, яка обчислює витрати послідовності дій. Ця функція витрат діє як система винагороди. Якщо агент поводиться правильно, витрати знижуються, якщо він поводиться небажано, витрати зростають. Важливо, щоб функція витрат враховувала цілу послідовність дій, а не одну, оскільки дорогі дії можуть призвести до зниження витрат у майбутньому. Складною частиною цього типу алгоритму є правильне відображення між розрахованими витратами та окремими діями. Важко ідентифікувати дії, які відповідають за високі витрати, і дії, які відповідають за низькі витрати. Іншою складною частиною цього алгоритму є визначення самої функції витрат. Цю функцію слід вибирати ретельно, оскільки вона безпосередньо визначає успіх чи невдачу програми [5].

Процес виявлення шаблонів у вхідних даних називається некерованим навчанням [7]. Для цього типу алгоритму не надаються бажані вихідні значення, і агент не може спостерігати за своїм середовищем. Тому єдина можливість навчитися - це переглянути дані, які передаються агенту, і визначити, чи можна розділити їх на якісь класи. Важливо відзначити, що агент, який навчається абсолютно без нагляду, не може визначити, як діяти, оскільки він не в змозі сприймати наслідки своїх результатів і не має інформації про те, що є ціллю, а що ні.

1.2 Основні методи еволюційного глибокого навчання для розробки програмних засобів автоматизованого управління агентом

Глибоке навчання як перспективна технологія широко використовується для вирішення різноманітних складних завдань, зокрема і в розробці програмних засобів управління в робототехніці для аналізу зображень [58] і розпізнавання образів [59].

Сучасні методи глибокого навчання полягають у ручній розробці глибоких моделей та знаходження відповідних конфігурацій методом проб і

помилок. Приклад наведено на рисунку 2, де знання предметної області передаються до глибокого навчання на різних етапах, таких як розробка функцій [60], генерація моделі [61] і розгортання моделі [62, 63]. На жаль, обмеженість доступу до експертних знань часто ускладнюють розвиток глибокого навчання.

Натомість, автоматизоване проектування глибоких нейронних мереж має тенденцію домінувати останніми десятиліттями [64, 61]. Основна причина полягає в гнучкості та ефективності обчислень автоматизованого машинного навчання у розробці функцій [60], оптимізації параметрів [65], оптимізації гіперпараметрів [66], пошуку нейронної архітектури [67, 68, 61] та стиснення моделі [69]. Саме тому автоматизоване машинне навчання без ручного втручання привертає значну увагу та досягає значного прогресу.

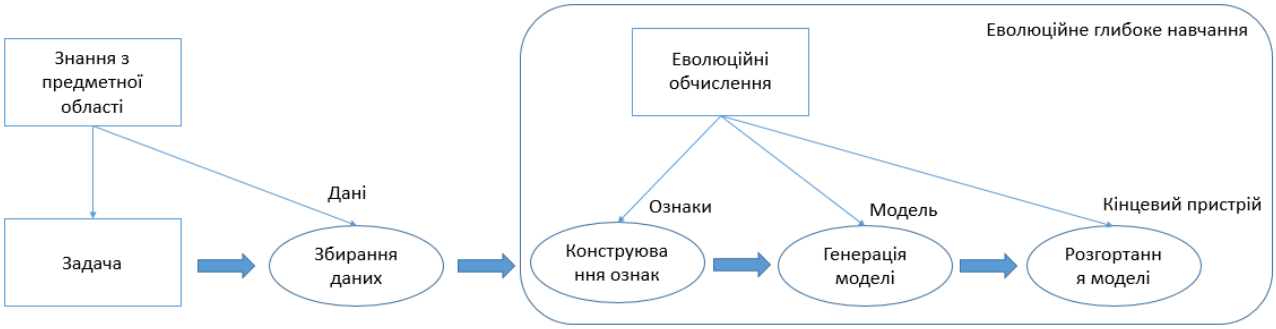


Рис. 2 Огляд глибокого навчання на основі знань предметної області або еволюційних обчислень

Еволюційні обчислення широко застосовуються для автоматизованого глибокого навчання завдяки своїй гнучкості та механізму, що автоматично розвивається. У еволюційних обчисленнях популяція індивідуумів керується відбором середовища, розвиваючись до оптимального рішення [70]. В даний час існує багато методів автоматизованого глибокого навчання на основі еволюційних обчислень, які називаються еволюційним глибоким навчанням [71, 72, 73, 74]. Наприклад, було проведено низку досліджень еволюційних обчислень щодо розробки функцій [60], генерації моделей [68, 61] та розгортання моделей [63]. Таким чином, інтеграція еволюційних обчислень та

глибокого навчання стала актуальною темою досліджень як в академічних, так і в промислових спільнотах.

1.2.1 Глибоке навчання

Глибоке навчання можна описати як трійку $M = (D, T, P)$ [68], де D – це набір даних, що використовується для моделі глибокого навчання (M), а T – цільове завдання. P вказує на продуктивність M. Метою глибокого навчання є підвищення продуктивності для конкретного завдання T, яке вимірюється P на наборі даних D. Далі представлені три фундаментальні процеси глибокого навчання: розробка функцій, генерація моделі та розгортання моделі.

Розробка функцій спрямована на пошук високоякісного набору даних D для покращення продуктивності P моделі глибокого навчання M для конкретних завдань T. На практиці простір функцій D може містити надлишкову та зашумлену інформацію, яка шкодить продуктивності (P) моделі (M). Наприклад, у медичному наборі даних Prostate dataset розмір піднабору ознак (65), вибраного в [75], становить лише 1% від загального розміру ознак (10509).

Генерація моделі має на меті оптимізацію або генерацію моделі M з бажаною продуктивністю (P) для конкретного завдання (T) на заданих наборах даних (D) [67]. Генерацію моделі можна розділити на оптимізацію параметрів, оптимізацію архітектури моделі та спільну оптимізацію [61]. Оптимізація параметрів полягає в пошуку найкращих параметрів (наприклад, ваг) для попередньо визначеної моделі. Оптимізація архітектури спрямована на пошук оптимальної топології мережі (наприклад, кількості шарів і типів операцій) моделі глибокого навчання (M) [76]. Спільна оптимізація пов’язана з двома вищезазначеними проблемами оптимізації шляхом автоматичного пошуку потужної моделі (M) в наборах даних (D) [77].

Розгортання моделі спрямоване на розгортання моделі глибокого навчання (M) для вирішення завдання T з прийнятною продуктивністю (P) на

вхідних даних (D) у межах обмежених обчислювальних ресурсів. Ключовим питанням розгортання моделі є те, як зменшити затримку, обсяг пам'яті та енергоспоживання, коли кількість параметрів моделі глибокого навчання є значною. Наприклад, Transformer-XL Large має 257 млн параметрів [78].

1.2.2 Еволюційні обчислення

Еволюційні обчислення — це методи штучного інтелекту, які використовують біологічні та природні процеси для вирішення складних проблем. Для опису цього сімейства алгоритмів використовується слово «еволюція», оскільки як основу використовують теорію природного відбору.

Базуючись на теорії природного відбору Чарльза Дарвіна, що описана у його книзі «Про походження видів» (1859, Джон Мюррей), основою еволюційних обчислень є концепція моделювання індивідуума або популяції індивідуумів у системі для пошуку найкращого. Мета полягає в тому, щоб вивести або розвинути індивідуум, який може виживати та процвітати в такому штучному середовищі, дозволяючи йому змінюватися. Цей механізм зміни індивідууму відрізняється залежно від методу еволюційних обчислень, але в усіх випадках існує механізм, який кількісно визначає, наскільки добре індивідуум виживає.

Термін, що використовується для кількісної оцінки того, наскільки добре індивідуум може вижити або розвиватися, називається допасованістю. Це універсальний термін, який використовується в еволюційних обчисленнях та визначає, наскільки добре індивідуум може вижити чи існувати в середовищі. Допасованість можна виміряти багатьма способами, але в усіх випадках вона є головним визначальним фактором того, наскільки ефективний індивідуум чи популяція індивідів у вирішенні поставленої задачі.

Концепції природного відбору та допасованості були використані як основа кількох обчислювальних методів, розроблених для відтворення біологічного процесу репродукції. Деякі з цих методів навіть імітують генетичний мітоз у клітинах, який відбувається під час поділу хромосом і

спільного використання ДНК. Нижче наведено короткий перелік найбільш відомих алгоритмів еволюційних обчислень:

- *Штучне життя*. У цьому алгоритмі процеси імітують штучний процес самого життя за допомогою агентів («гра в життя» Конвея та клітинний автомат фон Неймана). Агенти часто переміщуються, живуть або гинуть залежно від їх близькості до інших агентів або середовищ. Хоча моделювання агентів часто виконується для імітації реального світу, його також можна використовувати для оптимізації процесів.
- *Диференціальна еволюція*. Процес, у якому пошук оптимізується шляхом поєднання диференціального числення з еволюційними алгоритмами. Ця техніка часто поєднується з іншим методом еволюційного обчислення, наприклад штучним життям. У цьому алгоритмі агенти розвиваються або змінюються, беручи різниці векторів і повторно застосовуючи їх до популяції.
- *Еволюційні алгоритми*. Ширша категорія методів еволюційних обчислень, які застосовують до задачі еволюцію у формі природного відбору. Ці методи часто зосереджуються на моделюванні популяції особин.
- *Еволюційне програмування*. Спеціалізована форма еволюційних алгоритмів, які створюють алгоритми за допомогою коду. У цих методах окремий індивідуум представлений блоком коду, і його відповідна допасованість вимірюється до деякого оптимального значення, згенерованого під час виконання коду. Існує кілька способів реалізації генерації коду для еволюційного програмування, і в багатьох використовуються більш конкретні методи, такі як експресія генів.
- *Генетичний алгоритм*. Цей метод використовує клітинний мітоз низького рівня, який спостерігається в організмах та дозволяє передавати генетичні ознаки нащадкам. Генетичний алгоритм — це моделювання цього процесу шляхом кодування характеристик індивідуума в послідовність

гена, де ця довільна послідовність гена, яка може бути такою ж простою, як послідовність 0 або 1, оцінюється за певним показником допасованості. Ця допасованість використовується для моделювання процесу біологічного відбору та спарювання батьківських особин для отримання нового комбінованого потомства.

- *Генетичне програмування*. Цей метод створює програмний код за допомогою генетичного алгоритму. У генетичному алгоритмі риси індивідуума є більш загальними, але в генетичному програмуванні ознака або ген може представляти будь-яку кількість функцій або іншу програмну логіку. Генетичне програмування — це спеціалізована техніка, яка дозволяє розробляти новий алгоритмічний код. Приклади цього використовувалися для написання коду симуляції агента, який міг знаходити вихід з лабіринта або створювати зображення.
- *Програмування експресії генів*. Цей алгоритм є подальшим розширенням генетичного програмування, яке розробляє код або математичні функції. У генетичному програмуванні код абстрагується до функцій високого рівня, тоді як у програмуванні експресії генів метою є розробка конкретних математичних рівнянь. Ключовою відмінністю між програмуванням експресії генів і генетичним програмуванням є використання дерев виразів для представлення функцій. У той час як у генетичному програмуванні дерева виразів представляють код, у виразах програмування експресії генів вони представляють математичне дерево виразів. Перевага полягає в тому, що код слідуватиме чітко визначеному порядку операцій на основі розміщення.
- *Оптимізація рою*. Це різновид штучного життя, який представляє собою моделювання штучних і певною мірою інтелектуальних елементів. У цьому алгоритмі оцінюється допасованість кожного елемента рою, і найкращий елемент стає центром для решти елементів.
- *Ройовий інтелект*. Цей алгоритм є методом пошуку, який імітує поведінку рою комах або птахів, щоб знайти максимальні значення для

задач оптимізації. Це дуже схоже на оптимізацію рою, але відрізняється в реалізації залежно від оцінки допасованості.

На рисунку 3 показано ієрархію методів еволюційного обчислення, які використовуються для еволюційного глибокого навчання. Інші методи еволюційного навчання можуть бути використані для покращення моделей глибокого навчання.

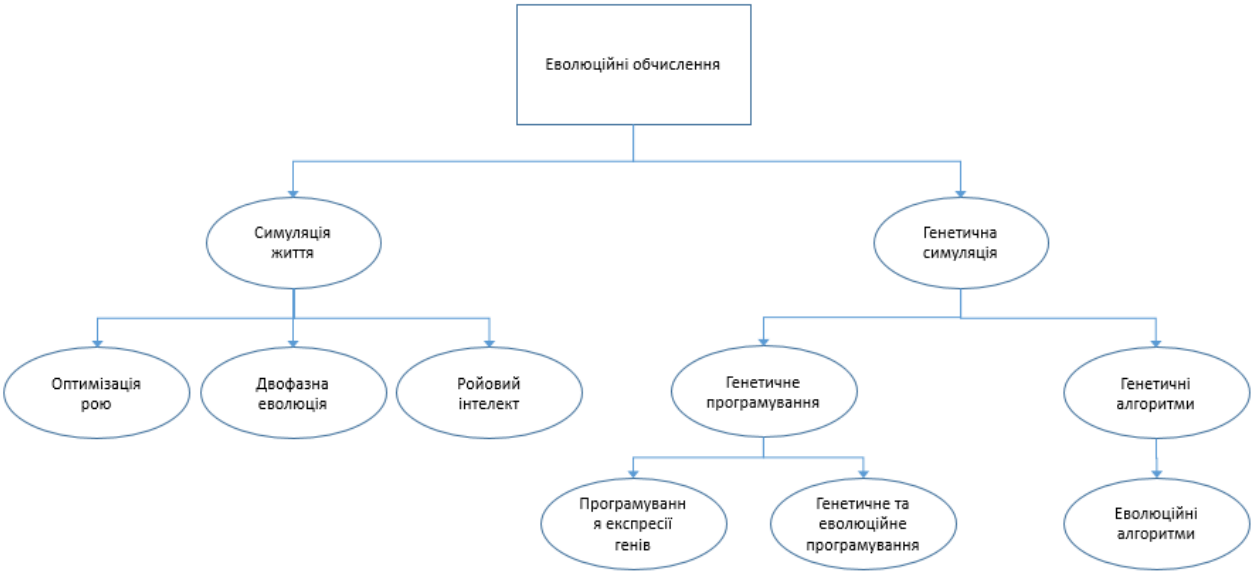


Рис. 3 Методи еволюційного обчислення для еволюційного глибокого навчання

Отже, еволюційні обчислення — це набір стохастичних методів пошуку на основі популяції, що базуються на механізмах еволюції, таких як природний відбір і генетика, що не потребують градієнтної інформації та здатні обробляти задачу оптимізації як чорну скриньку без явних математичних формулювань [79, 80]. Завдяки вищезазначеним характеристикам еволюційні обчислення широко використовуються для автоматизованого проектування у глибокому навчанні.

Загалом всі розглянуті вище методи еволюційних обчислень можна розділити на дві категорії: еволюційні алгоритми і ройовий інтелект [61]. Всі вони відповідають загальній структурі, як показано на рисунку 4, яка складається з трьох основних компонентів.

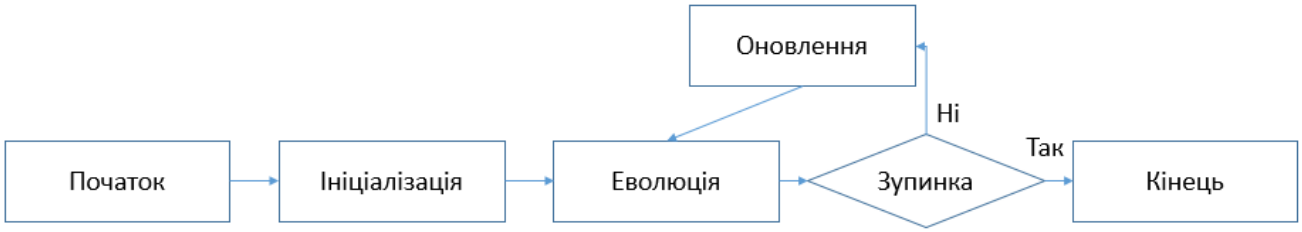


Рис. 4 Загальна схема еволюційних обчислень

Ініціалізація виконується для створення популяції індивідів, які кодуються відповідно до простору рішень (або простору пошуку та простору змінних) оптимізаційної задачі, наприклад набору функцій, параметрів моделі та топологічної структури.

Оцінка спрямована на обчислення придатності окремих індивідів. Фактично, оцінка індивідів у еволюційному глибокому навчанні є обчислювально дорогим завданням [81]. Наприклад, робота [82] використала 3000 GPU-днів, щоб знайти бажану архітектуру.

Оновлення має на меті генерувати ряд нащадків шляхом різноманітних операцій репродукції. Наприклад, нове рішення генерується за допомогою формули швидкості та положення в оптимізації рою частинок [75]. З точки зору генетичного алгоритму, деякі оператори репродукції (наприклад, кросинговер і мутація) використовуються для створення нових особин [83].

1.2.3 Еволюційні методи для оптимізації глибокого навчання

Еволюційне глибоке навчання — це концепція, що включає набір інструментів і методів для оптимізації мереж глибокого навчання. Проте воно також представляє набір інструментів, які можна накласти на глибоке навчання або навіть замінити його.

Еволюційне глибоке навчання має широку сферу застосувань: для базової оптимізації гіперпараметрів, пошуку ваг нейромережі для безперервних рішень, балансування змагальних мереж у генеративних змагальних мережах і навіть заміни глибокого навчання з підкріпленням. Еволюційні методи надають можливість для подальшої оптимізації або покращеного рішення для будь-якої системи глибокого навчання. Проте еволюційне глибоке навчання потребує

обчислень і може бути непридатним для простіших систем. Однак, для складних або нових проблем, еволюція пропонує ряд підходів.

Глибоке навчання — це потужна та сучасна технологія, яка має не лише переваги, а й недоліки. Одним із таких є потреба розуміти та оптимізувати модель. Це процес, який може вимагати безліч годин анотації даних або налаштування гіперпараметрів моделі.

Майже в усіх задачах модель не може бути використана безпосередньо у вихідному вигляді, і тому виникає потреба оптимізувати різні аспекти системи глибокого навчання: від налаштування швидкості навчання до вибору функції активації. Оптимізація моделі мережі часто стає основною проблемою, і якщо це робити вручну, може потребувати значних зусиль.

Оптимізація мережі глибокого навчання може охоплювати широкий спектр факторів. Окрім звичайного налаштування гіперпараметрів, розробнику також потрібно звернути увагу на саму архітектуру мережі [88]. Коли в процесі додавання шарів або різних типів вузлів мережа стає складнішою, це буде безпосередньо впливати на процес поширення нею втрат або помилок. На рисунку 5 показано найбільш поширеніші проблеми, які виникають під час створення більш складних і великих систем глибокого навчання.

У великих мережах величину втрат потрібно розділити на дедалі менші компоненти, які врешті-решт наближаються до нуля. Коли ці компоненти втрат або градієнти наближаються до нуля, це називається проблемою зникаючого градієнта, яка часто пов’язана саме з глибокими мережами. І навпаки, компоненти також можуть стати надзвичайно великими, проходячи через послідовні шари, які посилюють відповідні вхідні сигнали. Це призводить до того, що компоненти градієнта стають все більшими, або так званими розривними градієнтами.

Обидві проблеми з градієнтами можна вирішити за допомогою різних методів, таких як нормалізація вхідних даних або за допомогою шарів. Спеціальні типи функцій шару, які називаються нормалізацією та виключенням, показані на рисунку 5. Ці методи додають до обчислювальної

складності та вимог до мережі, а також можуть явно згладжувати важливі та характерні особливості даних. Таким чином, для розвитку високої продуктивності мережі потрібні більші та різноманітніші навчальні набори даних. Нормалізація може вирішити проблеми зникнення та вибуху градієнта глибоких мереж, але в міру зростання моделей вони виявляють інші проблеми. У міру зростання здатність моделей обробляти, наприклад, більші набори вхідних даних і зображень, зростає. Однак це може спричинити побічний ефект, відомий як мережеве запам'ятовування, яке може статися, якщо вхідний навчальний набір замалий. Це відбувається через те, що мережа настільки велика, що може почати запам'ятовувати набори вхідних фрагментів або навіть цілі зображення чи набори тексту.

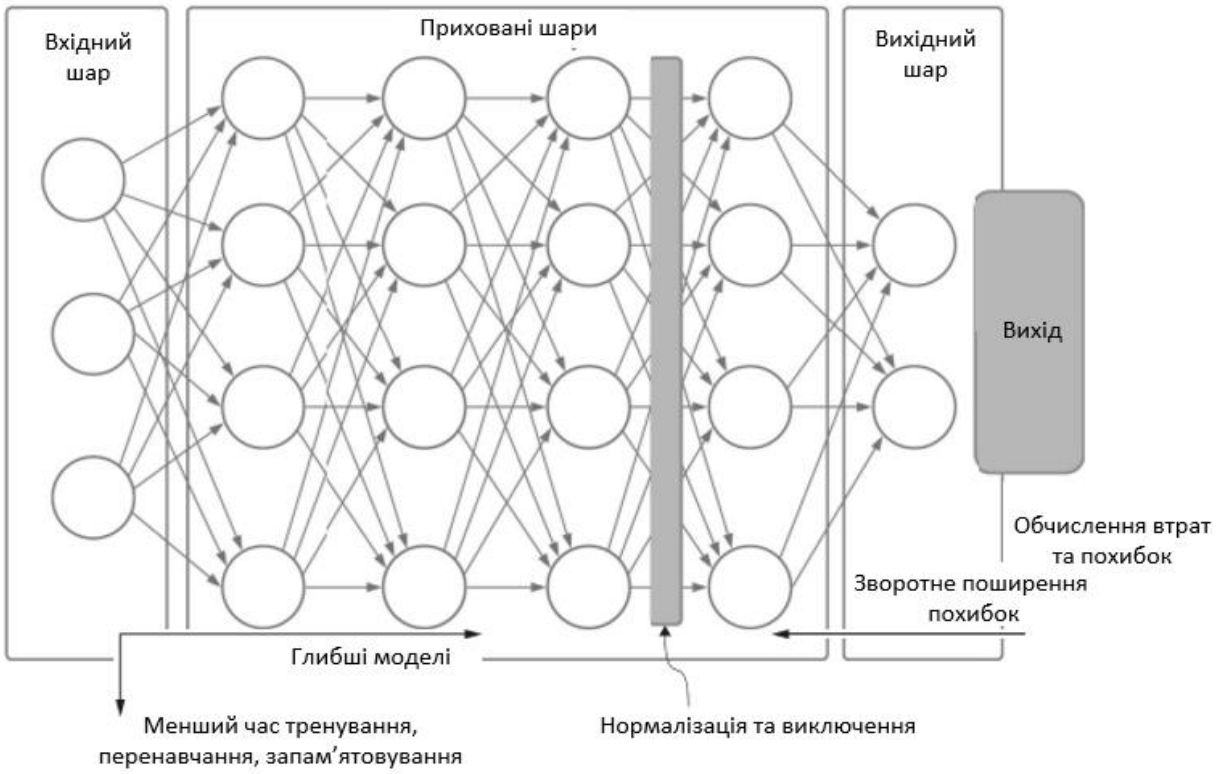


Рис. 5 Поширені проблеми розростання структури системи глибокого навчання

Передові моделі DL, наприклад GPT-3 (процесор природної мови від OpenAI), частково страждають від запам'ятовування. Ця проблема виникає навіть після того, як у такі моделі введено мільярди документів, що представляють різні форми тексту. Навіть з такими різноманітними та

масивними наборами тренувань, було показано, що моделі на зразок GPT-3 відтворюють цілі абзаци запам'ятованого тексту. Ця «проблема» може бути ефективною функцією для бази даних, яка погано вписується в модель DL. Були розроблені обхідні шляхи для вирішення проблеми запам'ятовування, яка називається «відпаданням», тобто процесом, за допомогою якого певний відсоток вузлів у мережевих рівнях може бути дезактивовано під час кожного проходу навчання. У результаті вимикання та вмикання вузлів протягом кожного проходу створюється більш загальна мережа. Однак це відбувається ціною збільшення мережі на 100–200%.

Окрім цих проблем, додавання більшої кількості рівнів для глибших мереж вимагає додавання більше ваг, які потрібно індивідуально тренувати протягом мільярдів і трильйонів ітерацій. Для навчання таких моделей потрібне експоненціальне зростання обчислювальної потужності, і багато найкращих, передових моделей зараз розробляються лише в організаціях, які можуть дозволити собі таку високу вартість.

Тенденція ширших і глибших мереж незабаром досягне плато для більшості розробок глибокого навчання, залишаючи будь-які майбутні передові розробки гігантам штучного інтелекту, таким як Google DeepMind. Тому простим рішенням є пошук альтернативних підходів, які можуть оптимізувати розвиток таких великих мереж. Для цього розглянемо застосування еволюційних обчислень до глибокого навчання для оптимізації архітектури мережі, вагових коефіцієнтів або всього разом.

Еволюційне глибоке навчання надає кілька потенційних методів, оскільки воно може автоматично оптимізувати розмір і форму мережі для різноманітних задач. Автоматична оптимізація є основою еволюційного глибокого навчання.

Оскільки еволюційні алгоритми забезпечують кілька шаблонів оптимізації, які можуть вирішити безліч задач, еволюційне глибоке навчання може бути використано в різних аспектах процесу розробки машинного

навчання, зокрема налаштування гіперпараметрів моделі для розробки даних або функцій, перевірка моделі, вибір моделі та архітектури.

1.3 Основні стратегії застосування еволюційного глибокого навчання

Еволюційні обчислення можна застосувати до глибокого навчання у кількох формах, які охоплюють різні автоматизовані стратегії в автоматичному машинному навчанні. На рисунку 6 показано різні підмножини еволюційного обчислення або еволюційного глибокого навчання, які можна застосувати до глибокого навчання, і де вони можуть застосовуватися в процесі розробки моделі штучного інтелекту.

Еволюційний метод	Симуляція життя	Генетичні алгоритми	Генетичне програмування	NEAT
Вибір моделі	Оптимізація параметрів	Оптимізація параметрів	Згідно шаблону	Оптимізація параметрів
Дизайн архітектури	Згідно шаблону	Оптимізація архітектури	Згідно шаблону	Оптимізація архітектури
Оптимізація гіперпараметрів	Оптимізація гіперпараметрів	Оптимізація гіперпараметрів	Згідно шаблону	Оптимізація гіперпараметрів
Валідація (помилки і втрати)	Згідно шаблону	Згідно шаблону	Оцінка помилок	Оцінка помилок

Рис. 6 Застосування еволюційних обчислень для процесу розробки моделі штучного інтелекту у глибокому навчанні

Вибір моделі: Пошук ваг

Обрана базова модель і типи шарів часто будуть залежати від задачі, що вирішується. У більшості випадків оптимізацію вибору моделі можна зробити швидко і вручну. Однак вибір моделі полягає не лише у виборі типу шарів; він

також може включати форму оптимізації, початкові ваги та втрати, які використовуються для навчання моделі. Шляхом оптимізації типів шарів моделі, механізму оптимізації та навіть форм втрат можна зробити мережу надійнішою для більш ефективного навчання.

Архітектура моделі: Оптимізація архітектури

Часто, будуючи мережі глибокого навчання, завищується розмір моделі або кількість вузлів і шарів у моделі. Потім з часом зменшується масштаб мережі, щоб вона стала більш оптимальною для вирішення задачі. У багатьох випадках наявність занадто великої мережі може призвести до запам'ятовування вхідних даних, що призведе до перенавчання. І навпаки, мережа, яка є надто малою для вивчення різноманітності та великої кількості даних, як правило, страждатиме від недонавчання.

Щоб вирішити проблеми з перенавчанням та недонавчанням, можна застосувати генетичні алгоритми для автоматичного зменшення мережі до мінімальної форми. Це не тільки покращує продуктивність моделі та обмежує перенавчання та недонавчання навчання, але також зменшує час навчання за рахунок зменшення розміру мережі. Це техніка добре працює при спробі оптимізувати глибші мережі.

Налаштування/оптимізація гіперпараметрів

Налаштування гіперпараметрів — це процес, який виконується в машинному навчанні та який вимагає оптимізації моделі шляхом налаштування різних управляючих змінних, які визначають цю модель. У глибокому навчанні параметри використовуються для позначення ваг моделі. Управляючі змінні мають назву гіперпараметри.

Еволюційні обчислення надають кілька альтернативних методів для додавання автоматичної оптимізації гіперпараметрів для широкого набору моделей, включаючи моделі глибокого навчання. Ройова оптимізація, диференціальна еволюція та генетичні алгоритми були успішно використані для ряду задач.

Валідація та оптимізація функції втрат

Розробка надійних моделей глибокого навчання часто базується на кількох встановлених шаблонах для створення якісних мереж. Цей процес може включати валідацію навчання та продуктивності моделі шляхом ітеративного перегляду втрат навчання та тестування, для того щоб переконатися, що обидва показники втрат не відрізняються дуже сильно.

У типовому сценарії навчання з учителем часто використовуються встановлені показники, які відповідають порівнянню міток. З більш просунутими генеративними сценаріями глибокого навчання стають доступними можливості для оптимізації форми втрат і навіть методів вілідації.

Мережеві архітектури, такі як автокодувальники, рівні вкладення та генеративні змагальні мережі, надають можливість використовувати комбіновані визначення втрат і валідацію моделі. З використанням еволюційних обчислень з'являється можливість використовувати методи автоматизованого машинного навчання для оптимізації цих типів мереж.

Нейроеволюція наростаючих топологій

Нейроеволюція наростаючих топологій (NEAT) — це техніка, яка поєднує оптимізацію гіперпараметрів і архітектури з пошуком ваг для автоматичного створення нових моделей глибокого навчання, які також можуть розробити свій метод втрат та валідації. Незважаючи на те, що NEAT було розроблено майже 20 років тому, лише нещодавно цю техніку почали застосовувати для різних застосунків з використанням глибокого навчання та глибокого навчання з підкріпленням.

1.4 Висновки до першого розділу

В даному розділі розглянуто основні методи розробки атоматизованого програмного управління роботизованими агентами як систем, які можуть бути представлені у вигляді взаємодії агента і середовища для виконання цільової задачі. Встановлено, що складність сучасних роботизованих систем вимагає програмного управління, яке виходять за межі традиційних методологій, що засновані на правилах. Інтелектуальне програмне управління на основі

штучного інтелекту, машинного навчання та нейронних мереж дозволяє отримати систему, що адаптується для виконання цільової задачі.

Як інструмент автоматизації розглянуто еволюційні методи та їх застосування для глибокого навчання моделей у програмних системах управління роботизованим агентом. На відміну від традиційного глибокого навчання, яке значною мірою покладається на знання експертів або предметної області для побудови моделі глибокого навчання, еволюційне глибоке навчання автоматично проектує глибоку модель через еволюційний процес.

З точки зору глибокого навчання, традиційне глибоке навчання потребує багато експертних знань для винайдення та аналізу інструменту навчання для конкретного набору даних або завдання. Навпаки, еволюційне глибоке навчання можна розглядати як зручний для розробника інструмент навчання, який може автоматично знаходити відповідні глибокі моделі на заданих наборах даних або завданнях. Отже, еволюційне глибоке навчання зосереджується на тому, наскільки легко можна використовувати інструмент навчання. З точки зору еволюційних обчислень, конфігурації моделі представлені як індивіди, а продуктивність як мета, яку потрібно оптимізувати. Еволюційні обчислення відіграють важливу роль в оптимізації, що базується на еволюційних механізмах. А еволюційне глибоке навчання можна розглядати як еволюційний процес оптимізації для пошуку оптимальних конфігурацій моделі глибокого навчання з високою продуктивністю. Нейроеволюція, еволюційна оптимізація гіперпараметрів і нейроеволюція наростаючих топологій є прикладами еволюційного глибокого навчання. Еволюційне глибоке навчання визначає підмножину методів еволюційного обчислення, які можна використовувати для автоматизації та покращення розробки моделей глибокого навчання на багатьох етапах робочого процесу машинного навчання.

Таким чином, сформулюємо ключові вимоги до розроблюваних методів автоматизації управління роботизованим агентом:

- підвищення адаптивності управляючої моделі до завдань навчання за допомогою підходу автоматичної розробки

- досягнення оптимальної моделі відповідно до розроблених цілей роботизованого агента або обмежень середовища
- використання нейроеволюційних методів, які поєднують оптимізацію гіперпараметрів і топології мережі з пошуком ваг для автоматичного створення нових моделей
- передбачення можливості розширення базових вхідних даних стану агента даними з камери, що забезпечить розробку програмного управління агентом, робота якого максимально наближена до умов реального світу

РОЗДІЛ 2. МЕТОДИ КОМП'ЮТЕРНОГО ЗОРУ ТА ПРОГРАМНІ ЗАСОБИ ДЛЯ УПРАВЛІННЯ РОБОТИЗОВАНОЮ КІНЦІВКОЮ З ВИКОРИСТАННЯМ ДАНИХ З КАМЕРИ

2.1 Основні методи використання зображень для задачі позиціонування мобільного роботизованого агента

Навігація агента за допомогою візуальної інформації полягає у постійному оновленні свого розташування та карти середовища шляхом здійснення вимірів відстані до оточуючих об'єктів. У техногенних умовах процес тривимірної реконструкції на основі декількох зображень може бути використаний, щоб допомогти роботам визначати їх положення у просторі та побудувати тривимірну карту навколишнього середовища [24].

Алгоритм одночасної локалізації та картографування (Simultaneous Localization and Mapping – SLAM) вирішує завдання оцінки позиції мобільного датчика та побудови карти навколишнього середовища в режимі реального часу. Монокулярний SLAM, що обробляє дані з однієї камери, став особливо актуальною темою досліджень останнього десятиліття. Малий розмір, мала вага та низький рівень енергоспоживання монокулярної камери роблять її підходящим сенсором для автономних роботів, таких як: автономні автомобілі, роботи-маніпулятори, безпілотні літальні засоби, системи доповненої реальності.

Одне з найважчих завдань монокулярного SLAM полягає у оцінці щільної карти досліджуваної сцени. Монокулярна камера надає інформацію з сенсорів у вигляді двовимірних зображень. Тому глибина кожного пікселя оцінюється з відношення координат точки реального світу між зображеннями з різних позицій камери. Такі відповідності виявляються шляхом порівняння фотометричних шаблонів на сусідніх пікселях кожного окремого пікселя. При використанні такого підходу виникають неточності: пікселі на малотекстурованих областях не можуть бути точно співставленні на

зображеннях та точна тривимірна реконструкція зазвичай обмежена областями зображень з великими градієнтами [24].

Сучасні SLAM-системи мають два основні компоненти: візуальна одометрія та глобальна оптимізація карти. У процесі роботи компонента візуальної одометрії (що поступово оцінює позицію камери та створює локальну карту) накопичуються невеликі помилки та з часом попередньо отримана позиція камери починає зміщуватись від дійсного розташування. Якщо виявлено попередньо відвідану локацію, зсув можна усунути за допомогою методів глобальної оптимізації карт, таких як закриття циклу шляху та оптимізація графу позицій. Хоча загальна цілісна карта середовища може бути отримана, накопичена помилка не може бути повністю вилучена з системи. Тобто, загальна ефективність будь-якої SLAM-системи визначається точністю модуля візуальної одометрії. Протягом останніх років, дослідження візуальної одометрії досягли значних результатів по підвищенню точності алгоритму, надійності та ефективності [25]. Здійснено спроби реалізації різних видів візуальної одометрії, наприклад, пряма та непряма (на ключових точках), щільна чи напівщільна оптимізація та розріджена оптимізація. Однак, незважаючи на таку різноманітність підходів, все ще залишається питання підвищення продуктивності під впливом якості фотометричного калібрування камери. Адже при розрахунках моделі візуальної одометрії неточні параметри камери використовуються на кожному етапі та не можуть бути скомпенсовані в результаті оптимізації.

Пікселі, що відповідають одній і тій же тривимірній точці, можуть мати різну інтенсивність на різних зображеннях внаслідок оптичного віньєтування камери, автоматичного підсилення та автоматичним налаштуванням експозиції. Існує декілька підходів до фотометричного калібрування з метою приблизного відновлення значень інтенсивності зображення. Останні дослідження довели, що фотометричне калібрування може значно покращити ефективність прямих методів візуальної одометрії [26].

На якість тривимірної реконструкції впливає декілька факторів: рух камери та об’єктів середовища, просторове квантування координат зображення [27], відповідність ключових точок [28], параметри калібрування камери [29], невраховані спотворення камери, а також числові та статистичні властивості обраного методу реконструкції [30]. Бхану та ін. [31] було продемонстровано, що реконструкція по зміні положень монокамери підлягає значним похибкам в околі фокусу розширення (focus of expansion - FOE), а реконструкція по стереозображенням має найбільшу похибку на краях поля зору (field-of-view - FOV). Хоча чисельні експериментальні результати дослідження показали, що дане твердження є справедливим, рішення цієї проблеми не було запропоновано. Для отримання аналітичних результатів, в роботі розглянуто алгоритм наближеної оцінки, а потім розраховується точність [32]. Такий підхід робить розрахунки простішими, проте використання субоптимального алгоритму є обмеженням, з огляду на накопичені втрати точності.

Дослідження різних типів помилок та їх вплив на якість тривимірної реконструкції, яка була отримана з використанням технології структурованого світла, представив у своїй роботі Ян та ін.[33]. Вчені сформували аналітичне представлення похибок, що виникають для положення тривимірної поверхні, її орієнтації та кривизни. Далі, Гажоуанті та ін. [34] запропонували підхід для оцінки границь похибок вимірювань, що враховують неточності, які виникають під час калібрування та тріангуляції. Баласураманян та ін.[35] проаналізували вплив шуму (який досі вважався незалежним і рівномірно розподіленим) та геометрії установки зображень на похибку реконструкції прямих ліній. Їх аналіз переважно базується на модельованих дослідженнях. Ревіра-Ріос та ін.[36] проаналізували помилку розмірних одиниць при вимірюваннях, ці помилки найчастіше обумовлені похибками локалізації на площині зображень.

Якість алгоритму реконструкції карти навколишнього середовища за допомогою методу SLAM в існуючих підходах була оцінена відповідно до точності отриманої траєкторії руху сенсора та швидкодії алгоритму. Якщо розглядати підходи SLAM з метою побудови точної щільної реконструкції

об’єктів навколишнього середовища, то необхідно враховувати три види похибок: похибки моделі камери, похибки при обробці зображень та похибки калібрування камери. На відміну від вищезгаданих досліджень, наш аналіз зосереджений на похибках результату фотометричного калібрування камери для дослідження залежності якості тривимірної реконструкції від точності фотометричного калібрування.

2.2 Основні методи програмного управління роботизованим агентом з камерою у середовищі

Кінематичне з’єднання [37] складається з набору твердих тіл (що називаються ланками), приєднаних один до одного через механізми (що називаються з’єднаннями), які обмежують їх рух. З’єднання мають від 1 до 6 степенів свободи, що визначають обмеження руху приєднаних ланок. З’єднання, яке поєднує ланку A та ланку B та має конфігурацію q_i відображає трансформацію: $T_B^A(q_i) \epsilon SE(3)$, і якщо q_i змінюється, то так само змінюється і трансформація між ланками A та B.

Перетворення будь-якої ланки L_i щодо стійкої системи відліку W може бути розраховано шляхом проходження кінематичним деревом та обчислення загальних перетворень, що ведуть від кожного з’єднання до кореня дерева (цей процес називається прямою задачею кінематики):

$$T_{L_i}^W = T_{L_i}^{L_{i-1}}(q_{i-1}) \dots T_{L_2}^{L_1}(q_1) T_{L_1}^W.$$

Конфігурацією робота $q \epsilon \mathbb{R}^{\mathbb{N}}$ є вектор, що поєднує степені свободи всіх зв’язків:

$$q = [q_1 \dots q_N]^T.$$

Часткова похідна i-тої системи відліку ланки по q має назву кінематичного Якобіана ланки, та для простих кінематичних ланцюгів може бути ефективно знайдено аналітичне рішення:

$$J_i(q) = \frac{\partial}{\partial q} T_{L_i}^W.$$

2.2.1 Датчики глибини зображення

Для відображення структури навколишнього середовища за допомогою системи комп'ютерного зору, робот повинен бути оснащений датчиками, що дозволять здійснити тривимірну реконструкцію середовища, а не лише двовимірне положення оточуючих об'єктів. Для цього може бути використана стерео камера або глибинна камера. Позначимо глибинне зображення I_D, як функція з областю визначення $\Omega \epsilon \mathbb{R}^2$. Відношення між тривимірними точками сцени та двовимірними точками на зображенні може бути змодельовано за допомогою простої пінхольної моделі камери:

$$Proj(x, y, z) = [u, v] = [\frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y],$$

де u, v - де двомірні координати на зображені; x, y, z - тримірні координати точок в системі координат камери та f_x, f_y, c_x, c_y – внутрішні параметри камери (фокусна відстань і центр камери - f ma c відповідно). Визначимо також обернену проекційну модель, яка приймає на вхід координати з камери u, v та вимір глибини точки z, та перетворює їх в тримірний вектор відносно фокусної точки камери. Отримана множина тримірних точок називається хмарою точок за глибинним зображенням. Для конкретного пікселя u, v з глибиною z, відповідна точка з хмари точок визначається наступним чином:

$$Proj^{-1}(u, v, z) = z[\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1].$$

2.2.2 Реконструкція середовища мобільним роботом.

Для вирішення задачі навігації мобільного робота використовується одночасна локалізація і картографування (ОЛК). Вона полягає у побудові і оновленні мапи невідомого оточуючого середовища з одночасним відстежуванням місцезнаходження під час руху у цьому середовищі.

При заданій послідовності зображень o_t з камери робота в дискретні проміжки часу t, задачею ОЛК є розрахунок і визначення положення робота x_t і мапи оточуючого середовища m_t: $P(m_t, x_t | o_{1:t})$.

Розглянемо реалізацію ОЛК методом CNN-SLAM [38]. Перевагою цього алгоритму є використання згорткових нейронних мереж для реконструкції по зображенням з монокамери. Точна та щільна тривимірна реконструкція досягається використанням прогнозованих карт глибини з глибинної нейронної мережі. Такі прогнозовані щільні карти глибини об'єднані зі значеннями глибини, отриманими з прямого монокулярного ОЛК. Схема методу зображена на рисунку 7.

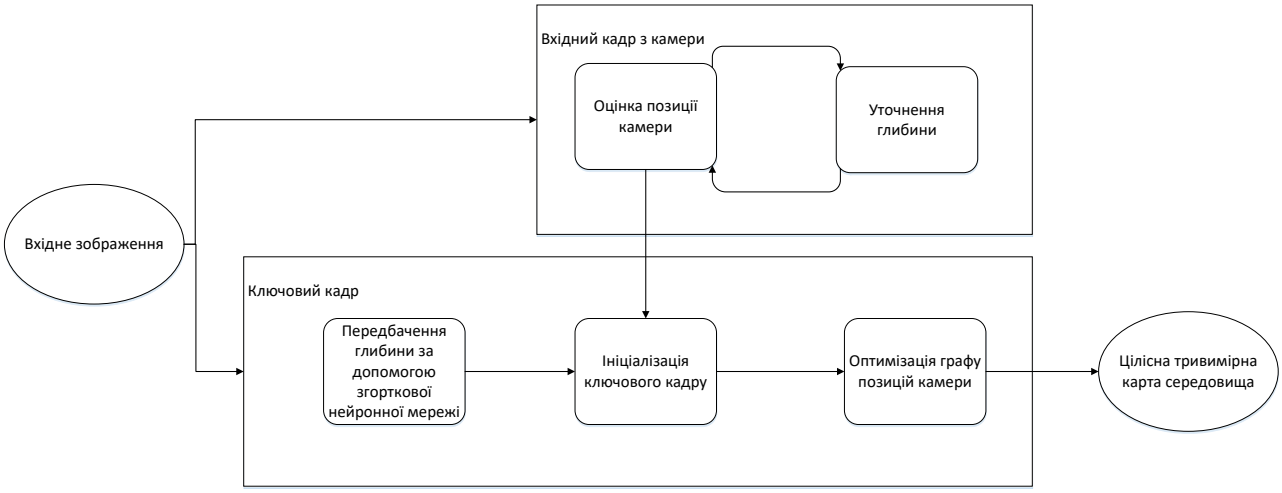


Рис. 7. Алгоритм CNN-SLAM

Алгоритм надає перевагу передбаченій глибині на частинах зображення, де підхід монокулярного ОЛК, як правило, не працює. Наприклад, вздовж мало текстурованих областей. Крім того, надається оцінка абсолютного масштабу отриманої реконструкції.

Оцінка положення камери відбувається шляхом визначення ключових кадрів $k_1, \dots, k_n \in \mathcal{K}$ як структурних елементів, на яких здійснюється ОЛК. Кожен ключовий кадр k_i пов'язаний з позицією камери T_{k_i}, щільною картою глибини D_{k_i} та картою невизначеності глибини U_{k_i} (степінь довіри кожному значенню глибини). Поточна позиція камери являє собою трансформацію t-го ключового кадру відносно t-1 ключового кадру і складається з поворотної матриці 3x3 $R_t \in \mathbb{SO}(3)$ та тримірного вектора переміщення $t_i \in \mathbb{R}^3$:

$$T_t^{k_i} = [R_1, t_1] \in \mathbb{SE}(3).$$

Ця трансформація оцінюється шляхом мінімізації фотометричної похибки між інтенсивністю зображення I_t поточного кадру та інтенсивністю зображення I_{k_i} найближчого ключового кадру k_i, застосовуючи зважену оптимізацію Гауса-Ньютона, що базується на функції:

$$E\left(T_t^{k_i}\right) = \sum_{\tilde{u}\epsilon\Omega} \rho\left(\frac{r(\tilde{u},T_t^{k_i})}{\sigma(r(\tilde{u},T_t^{k_i}))}\right),$$

де ρ - це норма Хьюбера та σ – це функціональний вимір залишкової невизначеності [38]. Фотометричний залишок r визначається на обмеженій підмножині пікселів, що лежать в областях великого градієнту кольору ($\tilde{u} \subset u \epsilon \Omega$) та визначається:

$$r\left(\tilde{u}, T_t^{k_i}\right) = I_{k_i}(\tilde{u}) - I_t(Proj(KT_t^{k_i}K^{-1}\tilde{u}D_{k_i}(\tilde{u}))).$$

Маючи значення $T_t^{k_i}$, поточну позицію камери у світовій системі координат можна обчислити як $T_t = T_t^{k_i}T_{k_i}$. Далі граф позицій камери на ключових кадрах уточнюються за допомогою методів оптимізації графів [39].

2.2.3 Детектування параметризованих поверхонь у тривимірній карті середовища

Після отримання тривимірної карти середовища у вигляді щільної хмари точок, важливою задачею є розпізнавання твердотілих об’єктів, що може бути здійснена на основі сегментації певної сцени середовища. Цей процес полягає у розподілі хмари точок на підмножини, що відповідають чітким параметризованим моделям, таким як: лінії, площини, кола, циліндричні поверхні, сферичні поверхні, конічні поверхні.

Сегментація хмари точок здійснюється на основі даних про нормалі точок з хмари. Карта, отримана на етапі реконструкції містить лише сукупність точок p_i, для кожної з яких оцінка нормалі представляє собою аналіз власних векторів та власних значень коваріаційної матриці C, що враховує найближчих сусідів заданої точки:

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p})(p_i - \bar{p})^T, C * \vec{v_j} = \lambda_j * \vec{v_j}, j\epsilon\{0,1,2\},$$

де k – кількість сусідніх точок, що розміщені у деякому околі точки p_i; $\bar{p}$ - центроїд найближчих сусідів; λ_j є j-им власним значенням коваріаційної матриці, а $\overrightarrow{v_j}$ - власним вектором. Отриманий вектор нормалі визначається:

$$\sigma = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2}.$$

Далі, на множині точок p_i та їх нормалей σ_i здійснюється сегментація з використанням методу RANSAC. Алгоритм знаходить різні параметричні моделі та представляє їх у формі кластеризованої хмари точок, де всі точки відсегментовані у екземпляри моделей об'єктів, яким вони належать.

2.2.4 Обчислення позиції здійснення захоплення цільового об'єкта

На основі цілісної щільної моделі об'єкта здійснюється планування захоплення. Для цього використано метод, що оцінює локальні властивості симетрії об'єкта [40]. Це може бути досягнуто шляхом обчислення серединної вісі, яка представляє тривимірний об'єкт як об'єднання куль. Дані симетрії, що містить серединна вісь, підлягають аналізу. Схему алгоритму представленого методу передбачення захоплення зображено на рисунку 8.

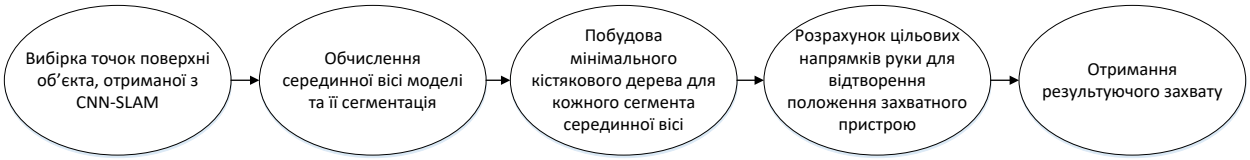


Рис.8. Планування захоплення на основі серединної вісі моделі об'єкта

Отримавши модель об'єкта довільної форми у вигляді хмари тривимірних точок, обчислюється серединна вісь за алгоритмом, представленим у [41]. Отриманий набір точок, що наближено утворюють серединну вісь, позначимо як M.

Дані серединної вісі на проекційній площині p представляємо як зважений повний граф та сегментуємо його на кластери c_i. Кожен утворений кластер c підлягає аналізу та відповідності певному типу серед наступних типів структур: коло, зірка з кільцем, дерево, елемент осі симетрії.

Генерування кандидатів на захоплення здійснюється з використанням набору евристик, що враховують отримані структури у частинах серединної

вісі. Основний підхід полягає у розгляді кількох сусідніх площин p_i та отриманні перспективних цільових напрямків та кутів повороту руки для кожного типу кластерів.

Показано, що інтеграція ОЛК системи з передбаченням карти глибини за допомогою глибокої нейронної мережі в систему розрахунку позиції руки робота для здійснення захоплення є багатообіцяючим напрямком для вирішення проблем розрахунку положень захоплюючого пристрою. Зокрема, вона дозволяє зменшити залежність від типових обмежень традиційної реконструкції по зображенням з монокамери, особливо щодо оцінки абсолютного масштабу, отримання щільної карти глибини на мало текстурованих областях і на кадрах, що мають лише трансформацію повороту. Запропонований підхід до використання уточнення карти глибини під час реконструкції навколишнього середовища долає ці проблеми, зберігаючи при цьому надійність і точність прямого монокулярного ОЛК. Загальна система здатна виконувати розрахунок положення захоплюючого пристрою на основі загальної реконструйованої сцени навколишнього середовища, її сегментації та виокремлення моделі цільового об’єкта [9].

2.3 Методи калібрування камери для зменшення впливу похибок даних із зображень

Камера, яка використовується як візуальний датчик, є невід’ємною частиною багатьох систем, серед яких робототехнічні, спостережувальні, дослідження космосу, промислова автоматизація. Для багатьох застосувань важливо знати параметри камери, щоб ефективно використовувати дані з зображень.

Процес оцінки параметрів камери називається калібруванням камери. Далі інформація про камеру (параметри або коефіцієнти) може бути використана для визначення точного співвідношення між тривимірною точкою в реальному світі та її відповідною двовимірною проекцією (пікселем) на зображенні, знятому цією відкаліброваною камерою.

Зазвичай калібрування полягає у знаходженні двох типів параметрів:

- Внутрішні параметри системи камера або об’єктиву, наприклад: фокусна відстань, оптичний центр і коефіцієнт радіального викривлення лінзи.
- Зовнішні параметри, які відносяться до орієнтації (повороту та зміщення) камери відносно певної світової системи координат.

Для статичної камери зовнішні параметри відповідають єдиному перетворенню: камера-світ. Однак, якщо камеру встановлено на рухомій руці робота, зовнішні параметри складаються з трансформації позиції камери позицію в робота та трансформації позиції робота в позицію в світовій системі координат. Трансформація позиції камери в позицію робота є функцією руху руки робота [90].

Зовнішні параметри є важливими для використання контрольованого руху камери в системі активного зору, оскільки рух робота зазвичай вказується в системі координат робота, тоді як 3D вимірювання отримують із вимірювань, зроблених щодо системи координат камери.

Складність оцінки внутрішніх параметрів полягає в тому, що класичні методи шукають рішення через деякі проміжні невідомі, які є складовими внутрішніх параметрів. Хоча метод найменших квадратів для оцінки проміжних невідомих є лінійним, процедура відновлення внутрішніх параметрів із проміжних невідомих не є такою. Невеликі помилки в оцінці проміжних невідомих викликають великі помилки в оцінці внутрішнього параметра. Однак, оскільки внутрішні параметри є інваріантними щодо розташування камери, їх оцінку можна покращити, використовуючи кілька переміщень камери. Більшість методів калібрування мають вирази замкнутого виду для рішення. Ці методи прості та ефективні, але можуть мати чисельну нестабільність через низький рівень зору для вилучення характеристичних точок зображення. Необхідно використовувати методи оптимізації для досягнення більш надійних результатів калібрування.

Точність вимірювань сцени залежить від складності прийнятої моделі камери. Найкращий результат досягається, коли використовується повна модель, включаючи нелінійні викривлення через недосконалості та зміщення оптичної системи. Розглянемо спершу модель пінхольної камери без корекції дисторсії. Дано тривимірну точку M, її зображення m є точкою перетину площини зображення та прямої, що проходить через M та оптичний центр C. Оптична вісь — це лінія, яка проходить через оптичний центр C і перпендикулярна до площини зображення. Система координат камери $X_C Y_C Z_C$ визначається так, що її початок є оптичним центром, вісь Z є оптичною віссю, а осі X і Y паралельні системі координат зображення. Отже, координата тривимірної точки (x_C, y_C, z_C) у системі координат камери та її координати (u, v) у системі координат зображення (u, v) пов'язані формулою

$$\begin{pmatrix} w_u \\ w_v \\ w \end{pmatrix} = \begin{bmatrix} \frac{f}{s_u} & 0 & u_0 \\ 0 & \frac{f}{s_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix},$$

де u_i — довільний скаляр, а елементи в матриці 3x3 складаються з внутрішніх параметрів камери: f — фокусна відстань камери, s_u і s_v — розмір пікселя по двум осям, а (u_0, v_0) — головна точка камери, тобто точка перетину оптичної осі та площини зображення. Оскільки фокусна відстань f не може бути відокремлена від розміру пікселя su або sv, калібрування розглядає співвідношення $k_u = \frac{f}{s_u}$ і $k_v = \frac{f}{s_v}$. Таким чином, необхідно оцінити чотири внутрішні параметри камери.

Далі розглянемо задачу калібрування рухомої камери. Нехай роборука з камерою може обертатися навколо вертикальної та горизонтальної осі. Ці рухи визначаються як чисті обертання навколо осей x і y системи координат робота. Камера жорстко прикріплена до робота, тобто взаємне положення та орієнтація камери та робота залишаються незмінними, коли робот переміщується. Для калібрування паттерн, розміщується в полі зору камери. Перетворення між двома системами координат, скажімо, W і C, описується однорідною матрицею

перетворення, H_{wc}, тобто координати тривимірної точки в системі C (x_C, y_C, z_C) з координатами (x_w, y_w, z_w) в системі W можуть бути отримані за допомогою наступного перетворення:

$$(x_C, y_C, z_C)^t = H_{wc}(x_w, y_w, z_w, 1)^t$$

$$H_{wc} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

де R = $\{r_{ij}\}$ і T = $(t_1, t_2, t_3)^T$ — матриця повороту та вектор трансляції, відповідно, із системи координат W до системи координат C.

Калібрування відбувається шляхом здійснення контрольованих рухів робота та спостереження за змінами в координатах зображення на калібрувальному патерні. На рисунку 9 показано зв'язки між різними системами координат, світом W, роботом G і камерою C, для двох місць розташування робота.

Щоб виконати калібрування камери, робот виконує набір відомих рухів, щоб отримати набір різних місць розташування камери. Тепер задачу, яку потрібно вирішити, можна сформулювати так: задано n (n > 7) опорних точок у світовій системі координат (x_i, y_i, z_i), (i = 1,…,n) та їхні координати на зображенні для набору різних місць розташування камер $(u_{ij}, v_{ij}), (i = 1,2,3, j = 1, .., n)$, і враховуючи відомі рухи робота $H_{g_1 g_2}, \dots, .H_{g_{n-1} g_n}$, оцінити внутрішні параметри f, k_u, k_v, u_0, v_0 , і зовнішні параметри камери H_{gc}, H_{wc1} (рисунок 9).

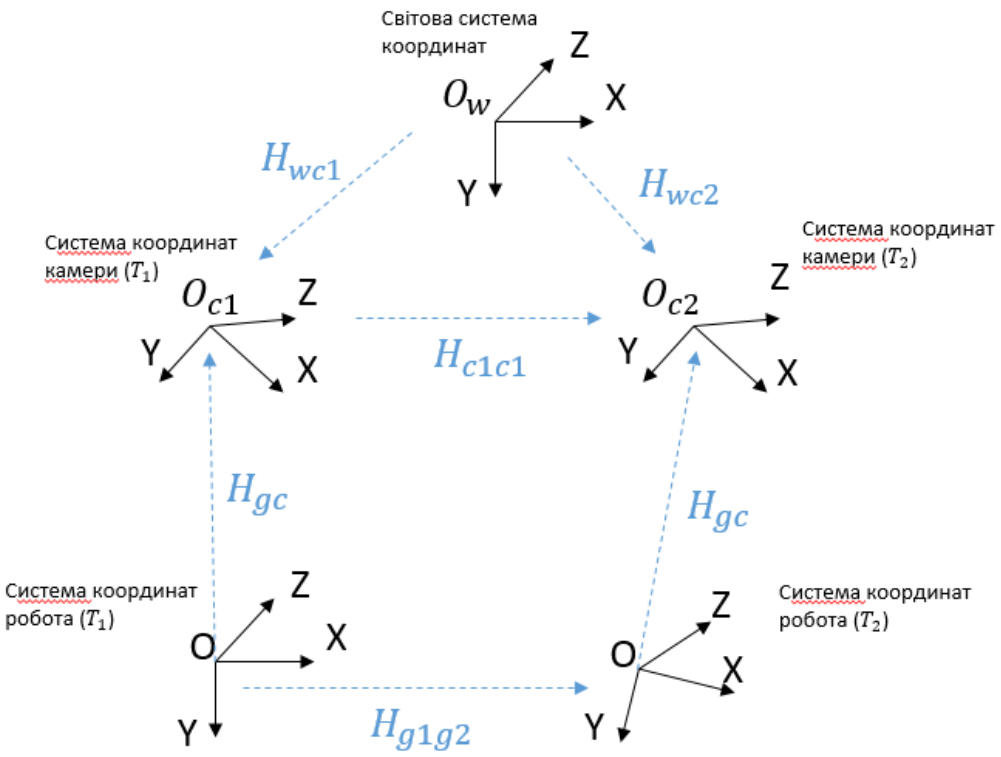


Рис. 9 Взаємозв’язок між системами координат для двох розташувань роборуки

Точність розв’язку щодо внутрішніх і зовнішніх параметрів залежить від кількох факторів: невизначеність координат ключових точок зображення; та нелінійність аналітичних методів для обчислення параметрів. Оскільки параметри відновлюються з проміжних невідомих нелінійно, невеликі помилки в оцінці проміжних невідомих можуть призвести до великих помилок в оцінках параметрів. Таким чином, бажано покращити оцінку за допомогою методів оптимізації.

Внутрішні параметри та трансформація між системами камери і робота незмінні для різних місць розташування камери. Таким чином, можна використати надлишковість даних, отриманих із набору місць розташування камери, для покращення оцінки цих параметрів. Цього можна досягти, безпосередньо оцінивши їх за допомогою методів оптимізації з початковим наближенням.

2.4 Оцінка впливу якості калібрування камери на точність системи з комп'ютерним зором

Для оцінки впливу похибок калібрування на тривимірну реконструкцію розроблено алгоритм обробки та аналізу сцени навколишнього середовища, що отримана в результаті роботи системи одночасної локалізації та картографування. Критерієм оцінки впливу обрано значення величини середньоквадратичного відхилення точок, що належать деякій заданій площині. Для цих цілей були розроблені модулі отримання зображення, реконструкції тривимірної хмари точок, детектування площини та розрахунку величини відхилення. Всі модулі були реалізовані на C++ Операційній системі роботів (Robot Operating System -- ROS), версія Indigo. Для процесів обробки вхідних зображень використано бібліотеку OpenCV [42]. Обробка від реконструйованої хмари точок здійснена з використанням бібліотеки PCL [24].

Модуль отримання зображень представляє собою зв'язок основної системи з камерою. Реалізований сервіс надсилає кольорові зображення у форматі, який підтримує ROS. Зображення, отримане з камери, є стиснутим для передачі з низькою пропускною спроможністю. Крім того, цей модуль також здійснює візуалізацію отриманих зображень.

У дослідженні використано камеру з об'єктивом «риб'яче око» з кутом огляду 166°. Для ректифікації вихідного зображення здійснюється процедура тривимірного калібрування. Вона представляє собою обчислення зовнішніх та внутрішніх параметрів камери.

Нехай P – тривимірна точка з координатами $\bar{X}$ у світовій системі координат. Вектор координат точки P у системі камери: $\overline{X_C} = R\bar{X} + \bar{T}$. Тут R – поворотна матриця, що відповідає вектору повороту om: R = rodrigues(om). Тоді маємо координати $\overline{X_C}$: $(\overline{X_C}_1, \overline{X_C}_2, \overline{X_C}_3)$ = (x, y, z).

Координати пінхольної проекції точки P(a,b) можуть бути представлені:

$$a = \frac{x}{z},$$

$$b = \frac{y}{z},$$

$$r^2 = a^2 + b^2.$$

Покладемо $\theta = \text{atan}(r)$. Тоді модель дисторсії [14]:

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8).$$

Координати спотвореної точки $P`$ представляють собою вектор $\bar{X}`(x`, y`)$:

$$x` = (\frac{\theta_d}{r})a,$$

$$y` = (\frac{\theta_d}{r})b.$$

Проекція точки P на зображення представляє собою точку $P_{Im}(u, v)$:

$$u = f_x(x` + \alpha y`) + c_x,$$

$$v = f_y y` + c_y.$$

Для отримання калібрувальних параметрів камери необхідно зібрати набір зображень калібрувального шаблону в різних положеннях камери. Для даного дослідження використано набір обсягом 184 зображення. Декілька прикладів з набору наведено на рисунку 10 [24].

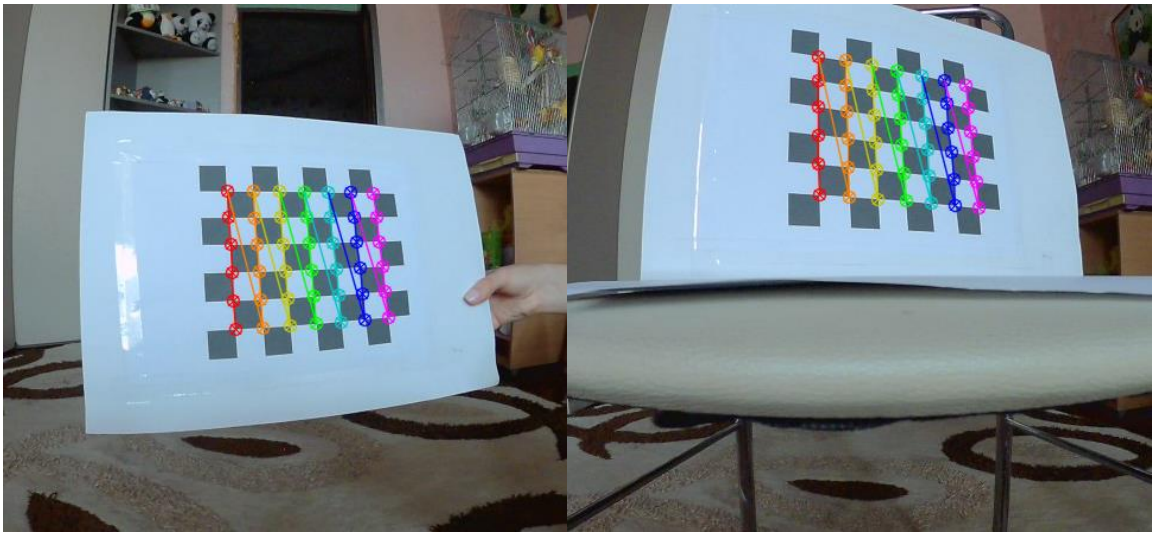


Рис.10 Приклади з калібрувального набору зображень

Застосовуючи методи бібліотеки OpenCV, отримуються внутрішні параметри камери та застосовуються до кожного вхідного зображення для усунення спотворень об’єктива.

Для отримання тривимірної інформації про сцену та оцінки поточного положення модуля камери, використано модуль на основі системи LSD-SLAM. Середня швидкість отримання нового ключового кадру в системі та оцінки

положення становить приблизно 5 та 10 Гц. Як показано на рисунку 11, оцінка глибини відбувається переважно на контурах на зображення. Такий результат є характерним для бібліотеки LSD-SLAM. Принцип її роботи полягає у пошуку різниці інтенсивностей зображень та знаходженні відповідностей на контурах текстур залежно від контрастностей сцени. Для оптимальної реконструкції сцени необхідно забезпечити її статичність. Даний модуль забезпечує збір та злиття від реконструйованих хмар точок з різних ключових кадрів (рисунок 11). Подальша оптимізація реконструкції може бути здійснена шляхом пост-обробки отриманої хмари точок [24].

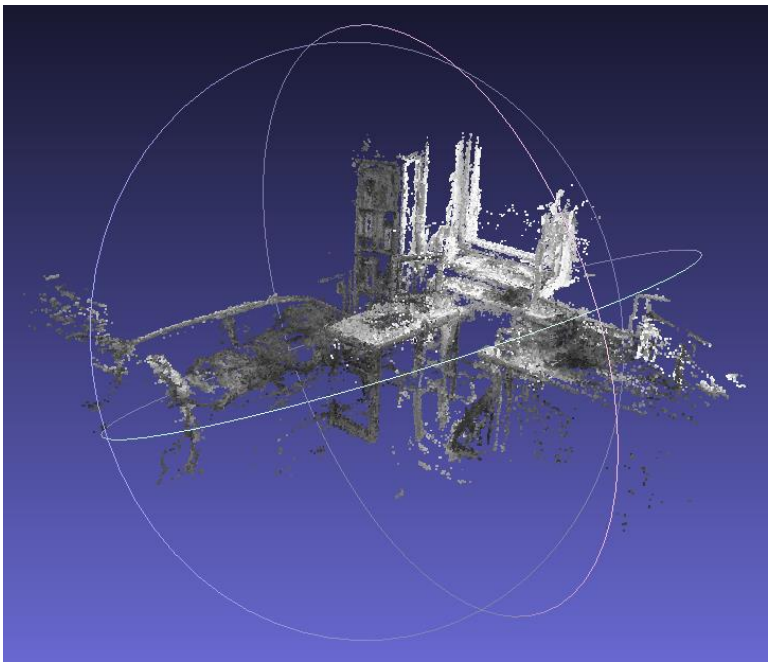


Рис. 11 Тривимірна модель у результаті реконструкції сцени

Для сегментації площини з хмари точок застосовано алгоритм RANSAC (RANdom SAmple Consensus) [43]. Цей метод оцінює параметри математичної моделі для набору спостережуваних даних які містять велику кількість викидів. Він випадковим чином вибирає мінімальний набір точок для оцінки параметрів моделі. З випадкових зразків він вибирає той, який найкраще відповідає повному набору точок. За своїм загальним формулюванням, RANSAC може бути легко застосований для опису будь-яких примітивних геометричних форм. Проте основний підхід RANSAC передбачає, що вхідні дані можуть належати лише одній моделі.

Функція пріоритетів з м’яким порогом [44], що базується на двох вагових функціях, використовується для поліпшення якості сегментації, яка враховує як відстань від точок до площини, так і узгодженість між векторами нормалі. Однак, це вимагає оцінки вектора нормалі в кожній точці, що є неефективним у хмарах точок з великою щільністю.

Відповідно до оцінок [45], часова складність RANSAC залежить від розміру підмножини, частки викидів та кількості точок в наборі. Час роботи RANSAC може бути надмірно довгим у деяких випадках. Тому розглядається модифікація алгоритму для більш ефективного детектування форм у хмарах точок – включаючи плоскі форми. Октодерево використовується для встановлення просторової близькості між зразками та їх функція оцінки враховує лише локальну підмножину зразків. Локальна вибірка шляхом відбору точок всередині кожного вузла, використовується для уникнення некоректних результатів.

Оцінку впливу похибок калібрування на тривимірну реконструкцію здійснено шляхом порівняння метрик для окремих площин при різному рівні внесеної штучно похибки та оцінки впливу величини похибки на ці метрики. Для цього розраховано величину середньоквадратичного відхилення хмари точок та на основі неї проведено оцінку площинності.

Маємо хмару точок, що представляє собою набір точок $P_i(X_i, Y_i, Z_i)$ в системі координат площини. Дана система координат є такою, що вісь z є перпендикулярною до площини. Перетворення, що переводить точку з глобальної системи координат реконструкції у локальну систему координат площини може бути представлено у вигляді: $P_L = R * P_G + T$.

Визначимо точку $O(\bar{X}, \bar{Y}, \bar{Z})$, таку що:

$$\bar{X} = \frac{\sum X_i}{n},$$

$$\bar{Y} = \frac{\sum Y_i}{n},$$

$$\bar{Z} = \frac{\sum Z_i}{n}.$$

Розглянемо точку O як центр координат нової локальної системи. Координати точок у новій системі можуть бути представлені:

$$x_i = X_i - \bar{X},$$

$$y_i = Y_i - \bar{Y},$$

$$z_i = Z_i - \bar{Z}.$$

Представимо площину у локальній системі координат як $z = ax + by$, де a та b можуть бути оцінені з наступних виразів (передбачаючи, що вимірювання відхилень здійснюється вздовж осі z):

$$a = \frac{\sum y_i^2 \cdot \sum z_i \cdot x_i - \sum x_i \cdot y_i \cdot \sum z_i \cdot y_i}{\sum x_i^2 \cdot \sum y_i^2 - (\sum x_i \cdot y_i)^2},$$

$$b = \frac{\sum x_i^2 \cdot \sum z_i \cdot y_i - \sum x_i \cdot y_i \cdot \sum z_i \cdot x_i}{\sum x_i^2 \cdot \sum y_i^2 - (\sum x_i \cdot y_i)^2}.$$

Обчислимо відхилення між виміряними точками та відсегментованою площиною:

$$e_i = \frac{(z_i - ax_i - by_i)}{\sqrt{a^2 + b^2 + 1}}.$$

Відхилення площинності (FD) можна визначити за сумою значень максимуму додатного локального відхилення (TP) та максимального значення модулю від'ємного локального відхилення (FP):

$$FD = |e^+{}_{max}| + |e^-{}_{max}| = TP + FP.$$

Повноту метрик (C1), правильність (C2) та якість (Q) для оцінки представленого методу виражають наступні представлення:

$$C1 = \frac{FD}{FD + TP},$$

$$C2 = \frac{FD}{FD + FP},$$

$$Q = \frac{FD}{FD + TP + FP},$$

де TP – кількість дійсних площин, що вірно визначені, FN – кількість площин, що є нерозпізнаними, FP – кількість невірно розпізнаних площин [24].

2.4.1 Оцінка впливу похибок калібрування на якість тривимірної реконструкції

Оцінку впливу рівня похибок калібрування на тривимірну реконструкцію протестовано на трьох наборах даних, що відповідають різним сценам. Основні параметри наборів даних представлені у таблиці 1 [24].

Таблиця 1. Опис наборів даних

Індекс	Довжина, м	Ширина, м	Висота, м	Кількість точок	Середня щільність точок на $м^3$
1	7	5.2	3.2	45361	389
2	8.5	9.4	2.6	38265	184
3	7.3	7.2	2.5	37472	285

Отримані тривимірні моделі мають достатню щільність. Це спричинено тим, що алгоритм LSD-SLAM, як прямий метод, використовує всю інформацію на зображені, включно з контурами. Як бачимо, це забезпечує високу точність та надійність в мало текстурованих середовищах з використанням однієї монокамери.

Для кожної сцени проведено сегментацію площин та виокремлено хмари точок, які являють собою набори вимірювань, що належать одній площині. Далі проведено оцінку параметрів математичної моделі кожної з площин. Приклад окремої хмари точок, що відповідає зашумленій моделі площини, наведено на рисунку 12 [24].

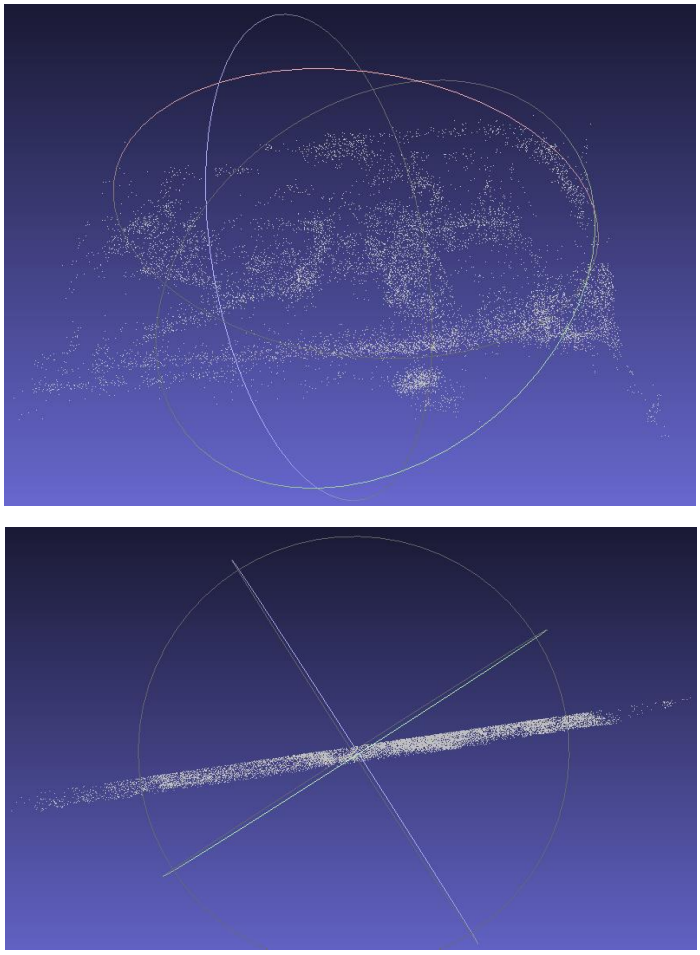


Рис. 12 Результат сегментації площини

Для кожної отриманої площини та відповідної їй хмари точок здійснено оцінку відхилень площинності та розраховано показники повноти метрик, правильності та якості. Далі досліджено вплив зміни параметрів калібрування на дані метрики. Для цього проведено аналіз чутливості параметрів камери, в якому здійснено спотворення значень пікселів на площині зображення шумом зі стандартним відхиленням від 0.05 до 1.0 пікселя. У таблиці 2 наведено вибіркові результати аналізу чутливості. У змодельованій системі камера розташовувалася таким чином, що напрямок вісі z глобальної та локальної систем координат співпадає [24].

Таблиця 2. Дисперсія параметрів калібрування як функція шуму

Доданий шум (пікс)	0.05	0.1	0.5	1.0
p_x (пікс)	1.93	4.25	17.63	38.97
p_y (пікс)	0.43	0.86	4.87	9.75
t_x (м)	0.00	0.00	0.03	0.03
t_y (м)	0.00	0.00	0.02	0.01
t_z (м)	0.00	0.00	0.03	0.02
R_x (град)	0.00	0.01	0.02	0.02
R_y (град)	0.06	0.14	0.76	1.48
R_z (град)	0.08	0.23	1.13	2.27

На основі отриманих зашумлених параметрів камери проведено процес реконструкції та сегментації площин з отриманих моделей. Для кожного такого набору вхідних даних проведено оцінку відхилень площинності. Залежність відхилення площинності від рівня внесеної похибки для кожного з трьох наборів даних представлено на рисунку 13.

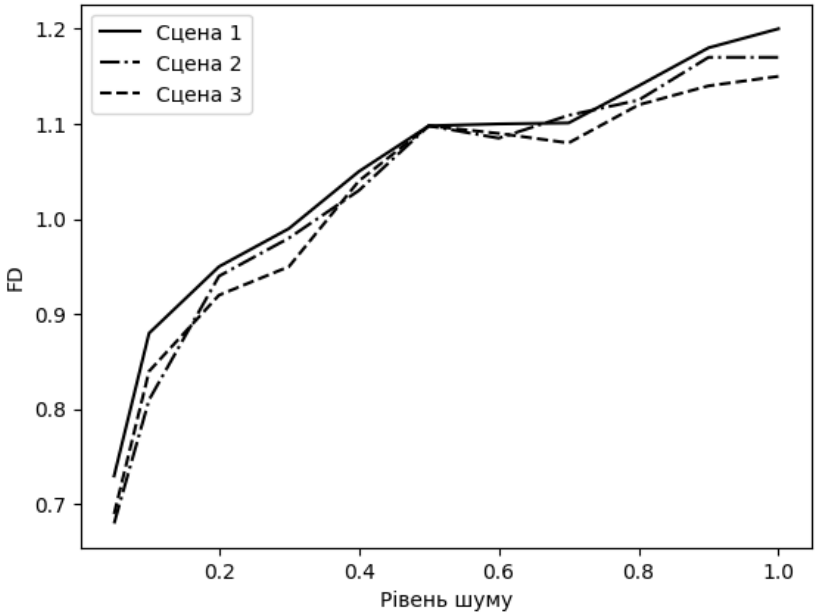


Рис.13 Залежність відхилення площинності від величини похибки параметрів калібрування

Даний експеримент продемонстрував залежність точності реконструйованої хмари точок від відхилень внутрішніх та зовнішніх параметрів камер у стереосистемі. Виявлено, що підвищення точності калібрування камер може надати можливість отримати приріст точності тривимірної моделі до 60% [56].

2.5 Оцінювання успішності захвату цільового об’єкту, здійсненого на основі зображень з камери

Розглянемо ефективність роботи системи тривимірної реконструкції на основі алгоритму одночасної локалізації та картографування як джерела моделі цільового об’єкту з точки зору успішності здійснення захватів моделей цільових об’єктів, що отримані даною системою.

Представлено систему побудови тривимірної реконструкції середовища на основі даних системи комп’ютерного зору робота для планування стійкого захвату цільового об’єкта. Розробка здійснена на основі таких принципів взаємодії компонентів: засобами системи відбувається обмін даними між підсистемами обробки вхідних зображень з камери, побудови тривимірної реконструкції середовища, отримання тривимірної моделі об’єкту та планування стійкого захвату робочою кінцівкою.

Механізм обміну даними реалізовано на основі системи ROS. Структура, створена під час дослідження, полягає в контролі емульованого захватного пристрою руки Jaco від одного "Master" комп'ютера. Це зроблено шляхом створення невеликої підмережі видавців ROS, що зв'язує емулятор руки разом з основною системою. Крім того, система пов’язана з монокамерою, яка закріплена на робочій кінцівці робота.

Поки емулятор обробляє дії руки за допомогою бібліотеки MoveIt!, основний комп'ютер може просто відправляти повідомлення і приймати інформацію про стан через обмін у відповідних темах.

Використання ROS значно спрощує інтеграцію між усіма компонентами, оскільки їх нормальне середовище роботи є неоднорідним.

Дійсно, емуляція руки системою MoveIt! контролюється окремим пакетом moveit_setup_assistant, тоді як обробка зображень з камери та побудова тривимірної моделі середовища запрограмовані за допомогою окремих бібліотек C++ (OpenCV та PCL відповідно). Таким чином, ROS дозволяє автоматизувати зв'язок між різними програмами, що працюють одночасно на одній машині та на інших машинах однієї і тієї ж під мережа.

Для оцінки розробленої системи обрано 4 моделі цільових об’єктів, що отримані з реконструкції сцени у формі хмари точок. Зокрема, обрано один об’єкт, який є занадто великим для здійснення захвату обраною кінцівкою. Алгоритм має розпізнати, що об’єкт не підходить та максимальний кут розкриття руки є надто малим.

Як бачимо, система побудови тривимірної моделі об’єкта та планування захвату для невідомих об’єктів демонструє хороші результати (таблиця 3). Всі тести проведено на персональному комп’ютері з процесором 3.2GHz на два ядра та середнім часом роботи близько 485.48 сек., планування оптимального захвату потребувало близько 385.67 сек. Для тестової хмари точок значення наведені у таблиці 3.

Таблиця 3. Тривалість обчислень

Обчислювальний етап	Час, сек.
Фільтрація хмари точок	14
Генерація мешу	57
Сегментація	3
Планування захвату	380
Загалом	454

Кожен з обраних об’єктів протестовано на чотирьох хмарах точок, як показано в таблиці 4. В жодному випадку робот не зміг здійснити захват об’єкта номер 2 через невідповідність його розмірів, які є занадто великими для даного захватного пристрою. Таким чином, незадовільний захват був успішно

протестований. Для того, щоб захват вважався успішним, рука робота повинна взяти об’єкт, стиснути його та тримати без впускання.

Таблиця 4. Успішність спланованих захватів тестових об’єктів

Номер	Об’єкт	Рівень успішності захвату
1	Коробка	100%
2	Кулька	0%
3	Чашка	85%
4	Іграшковий автомобіль	75%
Загалом		65% (87%)

В середньому, робот піднімав невідомий об’єкт у 65% випадків, включно з тестовим об’єктом номер 2, який не є задовільним. Якщо не враховувати 2 об’єкт, рівень успішності становить 87%. Для об’єкту у вигляді коробки-паралелепіпеда алгоритм відпрацював ідеально з рівнем успіху 100%. Хоча, об’єкти такі як чашка та іграшковий автомобіль є більш складними, адже містять кривизну та неоднорідність поверхонь, результати їх захвату є досить успішними з невеликим рівнем помилкових захватів [57].

2.6 Висновки до другого розділу

В даному розділі розглянуто методи комп’ютерного зору для програмного управління роботизованим агентом з захватною кінцівкою, що отримує інформацію про середовище з камери. Визначено основні підходи використання зображень для розробки програмного управління задач роботизованого агента та здійснено оцінку якості параметрів засобів комп’ютерного зору та впливу похибок параметрів камери на ефективність задачі захвату цільового об’єкту.

Застосовано метод оцінки точності моделі об’єкта для задачі стійкого захвату у комбінованій системі пропозиції захвату та реконструкції

тривимірної моделі об’єкта. Така комбінована система дозволяє здійснювати стійкий захват об’єктів будь-якої форми без обмежень на типи форм у тренувальному наборі даних. Класичні підходи реконструкції поверхні об’єкту базуються на відновленні глибини точок по парі зображень з двох камер. На якість тривимірної реконструкції впливає декілька факторів: рух камери та об’єктів середовища, відповідність ключових точок, параметри калібрування камери, невраховані спотворення камери, а також числові та статистичні властивості обраного методу реконструкції.

Визначено, що похибки параметрів камери можуть бути мінімізовані шляхом покращення процедури калібрування, тому досліджено вплив похибок на якість тривимірної моделі. Метрикою для оцінки точності обрано відхилення моделі від площини. Для її розрахунку здійснюється обробка хмари точок шляхом ідентифікації площин та сегментації, для яких розглянуто алгоритм, що базується на методі RANSAC.

Розроблено програмний засіб для здійснення оцінки точності реконструйованої моделі. Проведено експеримент для отримання залежності точності реконструйованих площин від похибок параметрів камер. Оцінку впливу похибок калібрування на тривимірну реконструкцію здійснено шляхом порівняння метрик для окремих площин при різному рівні внесеної штучно похибки та оцінки впливу величини похибки на ці метрики.

РОЗДІЛ 3. МЕТОДИ АДАПТИВНОГО НАВЧАННЯ ТА ПРОГРАМНІ ЗАСОБИ УПРАВЛІННЯ РОБОРУКОЮ НА ОСНОВІ НЕЙРОЕВОЛЮЦІЇ

3.1 Нейроеволюційний підхід для автоматизації розробки нейроконтролерів

Методи навчання та еволюційні техніки призначені для оптимізації швидкої реакції, яка потім використовується деяким контролером вищого рівня. Оскільки завдання та робоче середовище для роботів стають все більш складними, зростає потреба в методах навчання та пошуку, які можуть планувати досягнення мети, не покладаючись на вже існуючу структуру підзадач, розроблених людиною.

Оскільки традиційні методи глибокого навчання досить обмежені, все більше дослідників почали шукати альтернативні підходи до навчання штучних нейронних мереж. Глибоке машинне навчання надзвичайно потужне для розпізнавання образів, але не дозволяє виконувати завдання, які вимагають розуміння контексту або працюють з незнайомими даними. Багато дослідників сходяться на думці, що сучасний підхід до проектування систем штучного інтелекту вже не в змозі впоратися з актуальними проблемами.

Альтернативою традиційним методам глибокого машинного навчання є нейроеволюційні алгоритми. Нейроеволюція — це сімейство методів машинного навчання, які використовують еволюційні алгоритми для полегшення вирішення складних проблем, таких як ігри, робототехніка та моделювання природних процесів. Нейроеволюційні алгоритми імітують процес природного відбору. Кінцевим результатом нейроеволюції є оптимальна топологія мережі, яка робить модель більш ресурсоефективною та легшою для аналізу [47].

Нейроеволюційний алгоритм автоматично розвиває нейронні мережі для конкретного завдання та середовища. Перевагою є те, що необхідно лише абстрактно визначити бажану поведінку, а алгоритм максимально оптимізує

штучну нейронну мережу для виконання вимог. Однак застосувати нейронні мережі для робототехніки все ще складно, і досягнута продуктивність часто нижча, ніж очікується. Оскільки нейронні мережі загалом і нейроеволюційні алгоритми зокрема мають багато переваг і недоліків порівняно з альтернативами, цікаво розглянути їх порівняння з іншими підходами [48]. Дослідження показують, що все ще досить складно використовувати штучні нейронні мережі в задовільний спосіб для контролю стану автономного робота [49]. Щоб отримати прийнятні результати від генетичних алгоритмів, потрібно багато часу. Оскільки штучні нейронні мережі прямого зв'язку не мають жодного внутрішнього стану, їх обчислювальні можливості дуже обмежені, що може зробити їх гіршими від конкуруючих підходів. Однак нейроеволюція є абсолютно загальним підходом, який можна застосовувати без будь-яких детальних знань про навколишнє середовище. Крім того, цей клас алгоритмів може адаптуватися до непередбачених змін у середовищі [50]. При проектуванні контролерів спеціального призначення для досягнення прийнятних результатів потрібна детальна інформація про задачу та середовище. Крім того, важко або навіть неможливо розробити ці контролери таким чином, щоб вони могли адаптуватися до будь-яких змін у середовищі.

Таким чином, нейроеволюцію варто використовувати, якщо знання про задачу дуже обмежені або якщо задача може змінитися непередбачуваним чином у майбутньому. Проте, якщо середовище та задача добре визначені, написання алгоритму з нуля є кращим підходом. У таблиці 5 наведено огляд переваг і недоліків двох підходів.

Таблиця 5. Порівняння нейроеволюції та контролерів спеціального призначення [49]

Підхід	Переваги	Недоліки
Нейроеволюція	• Не потрібні детальні знання про проблему. • Агенти адаптуються до змін середовища	• Непередбачувані та суперечливі результати. • Симуляції з інтенсивними обчисленнями.
Контролери спеціального призначення	• Зазвичай більш висока продуктивність. • Результати передбачувані та зрозумілі.	• Необхідне детальне знання проблеми. • Зазвичай не в змозі адаптуватися до змін навколишнього середовища.

Дана робота присвячена розробці засобів автоматизації проєктування нейроконтролера для управління роботизованою кінцівкою. Точне розташування руки робота дозволяє точно та повторювано виконувати завдання, що призводить до підвищення ефективності та продуктивності. При правильному позиціонуванні роботи можуть виконувати завдання швидше, з меншою кількістю помилок і цілодобово, що значно економить час і кошти. Точне розташування кінцівки має важливе значення для виконання таких завдань, як обробка деталей, складання, зварювання або фарбування. Позиціонування руки робота має важливе значення для роботи в складних і динамічних середовищах. Точно розташовані руки робота можуть адаптуватися до мінливих умов, уникати перешкод і переміщатися складними шляхами. Це особливо важливо в таких галузях, як логістика, складське

господарство та охорона здоров’я, де роботам потрібно взаємодіяти з динамічним і незнайомим середовищем.

3.2 Програмні засоби для навчання нейроконтролерів роботизованих систем

Важливість автоматизованої розробки підкреслюється потребою в контролерах, які можуть легко адаптуватися до мінливого середовища, різних форм об’єктів маніпулювання і факторів, пов’язаних із середовищем. Навчальні можливості нейронних мереж, особливо з адаптивними структурами, роблять їх добре придатними для сценаріїв, де класичне програмування не може забезпечити знаходження мінімальної моделі, розмір якої є критично важливим для забезпечення заданого рівня ефективності.

Процес розробки контролерів на основі нейронних мереж для роборуки має набір складнощів. Починаючи від отримання та анотації різноманітних наборів даних і закінчуючи складним балансом між складністю моделі та налаштуваннями моделі. Дослідження еволюційних підходів для навчання адаптивних контролерів дозволить пришвидшити розробку контролерів для широкого спектру можливих сценаріїв використання кінцівки роборуки [23].

Програмне середовище для тренування агента-маніпулятора з використанням нейроеволюційного підходу має на меті автоматизацію процесу розробки, що є особливо актуальним у сценаріях, де ручне проектування та налаштування є непрактичним або займає багато часу.

Активний інтерес до дослідження та розробки систем, що базуються на навчанні з підкріпленням, дав поштовх для створення програмних середовищ, які дозволяють легку інтеграцію засобів для навчання моделі та інструментів для роботи з різноманітними агентами та середовищами.

Бібліотека Stable-Baselines3 [23] для навчання з підкріпленням на мові Python забезпечує набір високоякісних реалізацій різних алгоритмів навчання з підкріпленням. Вона створена на основі OpenAI Gym, що дозволяє легко використовувати її для тренувань агентів у різних середовищах, зокрема для

робототехнічних завдань. Бібліотека надає простий і зручний API для роботи з алгоритмами навчання з підкріпленням. Це робить її доступною як для початківців, так і для досвідчених користувачів. Крім того, вона пропонує різноманітні алгоритми навчання з підкріпленням, у тому числі такі популярні, як PPO (наближена оптимізація політики), DDPG (глибокі детерміновані градієнти політики) і TRPO (оптимізація політики довірчого регіону).

Stable-Baselines3 легко інтегрується з OpenAI Gym, набором інструментів для розробки та порівняння алгоритмів навчання з підкріпленням. Це дозволяє користувачам використовувати широкий спектр середовищ, у тому числі для задач маніпуляції з роботизованими робочими кінцівками. Бібліотека забезпечує добре оптимізовану реалізацію різноманітних алгоритмів, що робить її придатною як для досліджень, так і для практичних застосувань.

Проте, хоча Stable-Baselines3 підтримує низку алгоритмів, деякі з них краще підходять для дискретних просторів дій. Для просторів безперервної дії частіше використовуються такі алгоритми, як PPO та TRPO. Крім того, Stable-Baselines3 розроблено для навчання підкріплення з одним агентом. Якщо завдання передбачає мультиагентну взаємодію (наприклад, співпрацю кількох робототехнічних рук для маніпуляції одним об’єктом), інструментів бібліотеки буде недостатньо.

Важливою особливістю є підтримка Stable-Baselines3 фізичного симулятора MuJoCo, що широко використовується для проектування робототехнічних агентів, зокрема маніпуляторів та крокуючих ро-ботів.

Альтернативою Stable-Baselines3 є бібліотека TF-Agents з відкритим кодом, побудована на TensorFlow та розроблена Google. Вона забезпечує модульну та розширювану структуру для навчання з підкріпленням. TF-Agents підтримує сценарії як з одним, так і з кількома агентами та має реалізацію різних алгоритмів.

Ray RLlib — ще одна бібліотека для навчання з підкріпленням, створена на основі розподіленої обчислювальної системи Ray. Вона підтримує широкий

спектр алгоритмів і пропонує такі функції, як розподілене навчання та налаштування гіперпараметрів.

Для порівняння ефективності навчання контролерів роборуки на основі нейромереж доступні і інші бібліотеки, які мають схожий функціонал, але відрізняються набором алгоритмів навчання з підкріпленням. Серед них Garage, Tianshou, SLM Lab. Порівняння найбільш поширених бібліотек представлено в таблиці 6 [2].

Таблиця 6. Програмні бібліотеки для дослідження методів машинного навчання роботизованих агентів

Бібліотека	Використана бібліотека глибокого навчання	Мультиагентна підтримка	Зручність використання API	Підтримка середовищ OpenAI Gym	Підтримка еволюційних алгоритмів	Переваги
Satble-Baselines3	TensorFlow	Обмежена	Висока	+	-	Великий вибір алгоритмів RL, інтеграція OpenAI Gym
TF-Agents	TensorFlow	+	Середня	+	-	Модульність, підтримка сценаріїв з одним або кількома агентами
Ray RLib	TensorFlow, PyTorch	+	Середня	+	+ (Еволюційні стратегії)	Розподілене навчання, налаштування гіперпараметрів, різноманітне API
Garage	TensorFlow	+	Середня	+	+ (Генетичні алгоритми, CEM)	Розширена підтримка алгоритмів
Tianshou	PyTorch	+	Середня	+	+ (CMA-ES)	Модульність, широкий вибір алгоритмів
SLM Lab	TensorFlow	+	Середня	+	+ (CEM)	Модульність

Як бачимо, існуючі програмні рішення надають широкий вибір інструментів для дослідження алгоритмів навчання з підкріпленням для роботизованих агентів у визначеному середовищі. Проте більшість з них не

надає підтримки нейроеволюційних підходів, які надають можливість розробки адаптивних роботизованих систем, що є важливим для дослідження багатофункціональної системи, максимально наближеної до роботи з задачами реального світу [2].

3.3 Опис навчального середовища для задачі позиціонування роботизованої руки

Позиціонування роботизованої руки – це процес точного керування положенням і орієнтацією кінцівки роботизованої руки (інструмента або захватного пристрою) для виконання певних завдань. Це положення має вирішальне значення для взаємодії руки з об’єктами, маніпулювання ними та точного виконання різних операцій.

Роботизована рука зазвичай складається з кількох шарнірів, які забезпечують руку ступенями свободи (DOF). DOF представляє кількість незалежних параметрів, необхідних для опису конфігурації руки. Кожен суглоб дозволяє руці обертатися або переміщатися вздовж певної осі, дозволяючи руці рухатися в кількох напрямках. Позиціонування роботизованої руки базується на використанні систем координат для визначення положення та орієнтації руки. Найпоширенішою системою координат є декартова система координат, де положення визначаються координатами X, Y і Z. Орієнтація руки може бути представлена за допомогою кутів Ейлера, кватерніонів або матриць обертання.

Пряма кінематика передбачає визначення положення та орієнтації кінцевого ефектора на основі кутів або довжин з’єднань. З іншого боку, зворотна кінематика передбачає знаходження кутів або довжин з’єднань, необхідних для досягнення бажаного положення та орієнтації кінцівки.

Планування траєкторії передбачає створення плавного й оптимального шляху, яким рука робота буде рухатися під час переходу з одного положення в інше. Він враховує такі фактори, як уникнення перешкод, спільні обмеження та оптимізація шляху на основі таких критеріїв, як час, споживання енергії або

точність. Планування траєкторії гарантує, що рука рухається ефективно та безпечно, досягаючи бажаного положення

Для точного позиціонування роботизованих рук використовуються різні методи керування. При управлінні з відкритим циклом попередньо визначені команди надсилаються на руку без зворотного зв’язку, припускаючи, що рука рухатиметься згідно з інструкціями. Контроль із замкнутим контуром, також відомий як контроль із зворотним зв’язком, включає зворотний зв’язок датчика для постійного регулювання положення руки та коригування будь-яких відхилень.

Взаємодія з цільовим об’єктом потребує певного планування. Для успішної маніпуляції об’єктом завдання буде розділено на дві підзадачі з окремими контролерами. Перша підзадача полягає в тому, щоб перемістити кінцівку відносно близько до цільової позиції, уникаючи перешкод. Друга підзадача відноситься до прямого маніпулювання об’єктом, наприклад переміщення випадково розміщених об’єктів із початкового стану до заданої цільової конфігурації. Необхідність використання двох контролерів для цього завдання демонструє, що нейроеволюції легше розвинути реактивну поведінку, ніж довгострокову стратегічну поведінку [4].

Розглянемо першу визначену підзадачу – позиціонування кінцівки роботизованої руки за допомогою середовища OpenAI Gym із двовимірною роботизованою рукою з двома суглобами. Це середовище моделювання, призначене для забезпечення віртуальної платформи для тестування та розробки алгоритмів керування для двовимірної роботизованої руки [51]. Приклад стану навколишнього середовища представлений на рисунку 14.

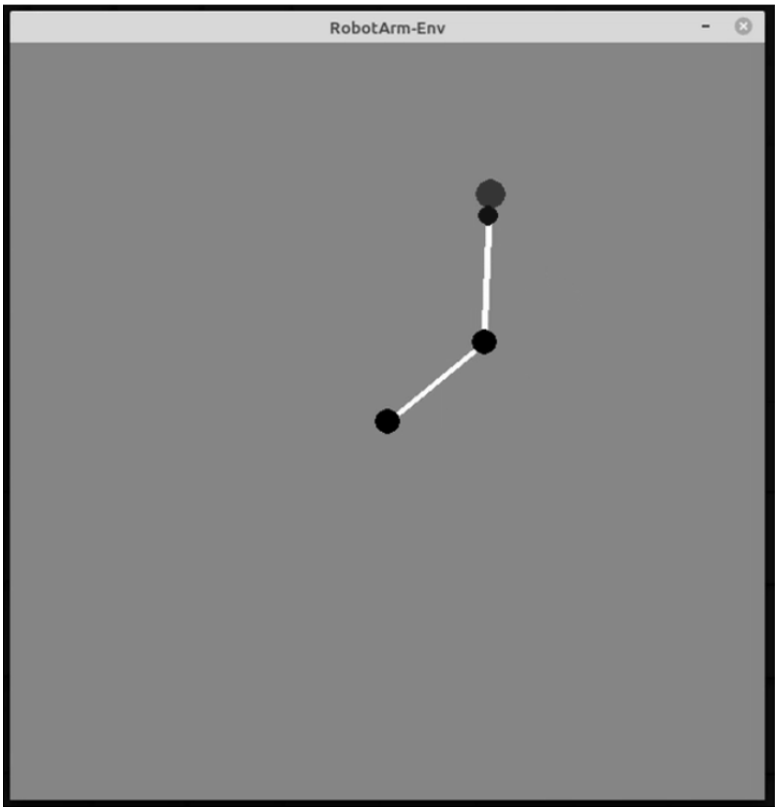


Рис. 14. Середовище OpenAI Gym 2D Robot Arm

Робот складається з двох з’єднань, кожне довжиною 100 пікселів, і мета — досягти червоної крапки, яка випадковим чином генерується в кожному епізоді. Простір дій представляє можливі дії, які може виконати агент. У двовимірному середовищі маніпулятора простір дій складається з керування кутами з’єднання маніпулятора робота:

• 0: залишити поточне значення кута з’єднання

• 1: приріст кута з’єднання 1

• 2: зменшення кута з’єднання 1

• 3: приріст кута з’єднання 2

• 4: зменшення кута з’єднання 2

• 5: приріст кутів з’єднань 1 і 2

• 6: зменшення кутів з’єднань 1 і 2

Простір спостережень дає інформацію про поточний стан навколишнього середовища. У середовищі двовимірної руки робота простір спостережень є безперервним і містить відповідну інформацію, необхідну для контролю та прийняття рішень:

• цільова позиція в напрямку x (у пікселях)

• цільова позиція в напрямку y (у пікселях)

• поточне положення з'єднання 1 (у радіанах)

• поточне положення з'єднання 2 (у радіанах)

Навколишнє середовище забезпечує винагороду для управління процесом навчання. Винагорода залежить від досягнення цільової позиції:

• робот отримає штраф -1, якщо поточна відстань між кінцівкою і цільовою позицією більша, ніж попередня відстань

• робот отримає винагороду 1, якщо поточна відстань між кінцівкою і цільовою позицією > -Ɛ < Ɛ, де Ɛ = 10 пікселів

Крім того, визначено умови завершення, щоб вказати кінець епізоду: поточна винагорода становить -10 або +10 [4].

3.4 Адаптивний підхід з використанням алгоритму NEAT

Навчання з підкріпленням (RL) успішно застосовано до політик контролю навчання для рук роботів. Алгоритми RL дозволяють роботам вивчати оптимальні стратегії керування методом проб і помилок, взаємодіючи з навколишнім середовищем і отримуючи зворотній зв'язок у формі винагород або штрафів [4].

У RL ми тренуємо політики агентів π_n, n = 1, ..., N для вирішення окремих випадків середовища: для кожного кроку середовища t агент спостерігає стан середовища s_t, і політика вирішує, яка дія a_t є має бути здійснена. Агент отримує винагороду $r(s_t, a_t, s_{t+1})$ за кожен спостережуваний стан під час епізоду навчання. Це призводить до траєкторії з T кроків. Сукупна винагорода $\sum_{k=0}^{T} \gamma^k r_{t+k+1}$ наприкінці епізоду, дисконтована за коефіцієнтом γ, зазвичай використовується як цільова функція політика процесу оптимізації. Називатимемо R_t повного епізоду пристосованістю політики [52].

Алгоритми RL навчаються через ітераційний процес взаємодії з середовищем. Рука робота виконує дії на основі поточної політики, спостерігає за кінцевим станом і винагородою та відповідно оновлює свою політику. Цей

процес триває, доки агент не наблизиться до оптимальної політики контролю, яка максимізує сукупну винагороду з часом. Навчання можна проводити онлайн, коли рука навчається під час взаємодії з фізичним середовищем, або офлайн, використовуючи попередньо зібрані дані чи симуляції. Використовуючи навчання з підкріпленням руки роботів можуть автономно навчатися політикам керування, які дозволяють їм виконувати складні завдання, адаптуватися до змінюваних середовищ і навіть вчитися на прикладі людей. Політики керування на основі RL були успішно застосовані до широкого діапазону задач, зокрема маніпулювання об’єктами, складання, захват та промислову автоматизацію.

Нейронна мережа обрана як політика керування роботом. Мережева архітектура зазвичай складається з вхідних нейронів, що представляють стан руки робота, прихованих нейронів для проміжних обчислень і вихідних нейронів, які генерують керуючі сигнали. Вагові коефіцієнти та зміщення нейронної мережі є параметрами, які потрібно оптимізувати. Під час кожного покоління нейронні мережі оцінюються на основі їх продуктивності в попередньо визначеній функції відповідності. Фітнес-функція вимірює, наскільки добре рука робота виконує певне завдання, наприклад досягає цілі, маніпулює об’єктами або уникає перешкод. Оцінка придатності може бути проведена шляхом моделювання або фізичних експериментів, залежно від наявних ресурсів [4].

Еволюційні алгоритми використовують різні оператори, такі як відбір, кросовер і мутація, щоб створити нові покоління нейронних мереж [46]. Відбір надає перевагу найкращим мережам, що дозволяє їм передавати свої гени наступному поколінню. Схрещування поєднує параметри двох батьківських мереж для створення нащадків із сумішшю їхніх ознак. Мутація вносить невеликі випадкові зміни в параметри, сприяючи дослідженню простору пошуку [53].

Процес нейроеволюції ітеративно розвиває популяцію нейронних мереж протягом кількох поколінь. Оцінка придатності, відбір, схрещування та мутація

повторюються, доки не буде знайдено задовільну політику контролю. Процес спрямований на збіжність до параметрів нейронної мережі, які надають політики керування з вищими значеннями допасованості, ефективно покращуючи продуктивність руки робота [54].

NEAT (NeuroEvolution of Augmenting Topologies) — це нейроеволюційний алгоритм, спеціально розроблений для розвитку штучних нейронних мереж (ANN) зі складною та змінною топологією. Він був представлений Кеннетом О. Стенлі та Рісто Мійкулайненом у 2002 році і з тих пір став популярним підходом у галузі нейроеволюції [53].

NEAT вирішує проблеми розвитку штучної нейронної мережі із різною кількістю нейронів і з’єднань. На відміну від традиційних методів еволюції нейронних мереж, які спираються на фіксовані топології, NEAT дозволяє створювати та розвивати нові мережеві структури.

Підхід використовує історичну схему маркування, щоб забезпечити збереження інновацій у структурі мережі під час процесу еволюції. Кожному гену в геномі присвоюється історичний маркер для відстеження його походження та збереження історичного запису інновацій. Це дозволяє зберігати та рекомбінувати нові структурні особливості протягом поколінь. Він включає видоутворення для заохочення збереження різноманітності в популяції. Під час еволюції особини групуються у види на основі їх подібності в структурі мережі. Видоутворення допомагає запобігти втраті перспективних структур через надмірну конкуренцію та дозволяє досліджувати різні топології.

Алгоритм дозволяє мережам рости та збільшувати складність протягом поколінь. Він поступово вводить нові вузли та з’єднання, починаючи з простих структур і поступово ускладнюючи їх у процесі еволюції. Крім того, NEAT підтримує спрощення мереж при виявленні зайвих або непотрібних з'єднань, покращуючи ефективність і продуктивність.

NEAT був успішно застосований у різних областях, включаючи системи керування, ігри, робототехніку та розпізнавання образів. Його здатність працювати з динамічними та змінними мережевими топологіями робить його

добре придатним для завдань, які потребують адаптивності, масштабованості та відкриття нових мережевих архітектур [4].

Розглянемо ключові особливості та концепції алгоритму NEAT.

- Генотип і фенотип: в NEAT індивідууми представлені як генотипом, так і фенотипом. Генотип кодує структуру нейронної мережі, включаючи вузли (нейрони) і зв'язки (синапси), тоді як фенотип представляє активну мережу з фактичними вагами, призначеними зв'язкам.
- Інновації та історичні маркування: NEAT використовує історичну схему маркування для відстеження походження генів (з'єднань і вузлів) протягом поколінь. Інновації в структурі мережі зберігаються, що гарантує, що нові функції не будуть втрачені під час еволюційного процесу.
- Ускладнення та спрощення: NEAT дозволяє нейронним мережам розвиватися та ускладнюватись протягом багатьох поколінь. Нові вузли та з'єднання можна вводити поступово, починаючи від простих мереж і розвиваючись до більш складних структур. Крім того, NEAT підтримує видалення зайвих або непотрібних структур, сприяючи спрощенню мереж, коли це виправдано.
- Видоутворення: NEAT включає видоутворення, процес поділу популяції на види на основі подібності. Це запобігає домінуванню кількох особин і дозволяє зберегти різноманітність. Специфікація заохочує дослідження різних мережевих структур і запобігає втраті багатообіцяючих інновацій.
- Спільний доступ до допасованості: NEAT використовує обмін допасованістю серед видів. Геноми з подібною мережевою структурою діляться значенями фітнес-функції, сприяючи співпраці всередині виду, а не інтенсивній конкуренції. Це гарантує, що інноваційні структури мають шанс розвиватися та вдосконалюватися.
- Схрещування і мутація: NEAT використовує операції схрещування та мутації. Схрещування поєднує гени двох батьків для створення нащадків із

сумішшю їхніх ознак. Мутація вносить невеликі зміни в гени, полегшуючи дослідження простору рішення.

Еволюційний цикл алгоритму NEAT передбачає створення нових поколінь нейронних мереж шляхом відбору, відтворення, кросинговеру та мутації. Придатність кожної окремої мережі оцінюється на основі її продуктивності в конкретному завданні. Протягом багатьох поколінь NEAT ітеративно розвиває мережі, які демонструють покращену продуктивність і підвищену складність, що призводить до рішень, які добре підходять для їхніх завдань.

Загалом, інноваційний підхід NEAT до розвитку нейронних мереж із різними топологіями вніс значний внесок у розвиток нейроеволюційних алгоритмів та їх застосування для оптимізації рішень, що базуються на використанні нейромереж [1].

Підхід поєднання еволюційних алгоритмів із нейронними мережами надихнув на подальші розробки в галузі нейроеволюції. Розширення та варіації NEAT були запропоновані для вирішення конкретних завдань, таких як мережі створення композиційних шаблонів (CPPN) для генеративних завдань або HyperNEAT для розвитку великомасштабних нейронних мереж. Загалом він виявився потужним і гнучким нейроеволюційним алгоритмом, що забезпечує еволюцію складних структур нейронної мережі та просуває сферу еволюційної робототехніки та штучного інтелекту [4].

3.5 Застосування методу оптимізації пошуком новизни для алгоритму NEAT

Хоча методи прямої оптимізації функцій допасованості добре працюють в багатьох простих задачах, при розв’язку більш складних задач вони часто потрапляють у пастку локального оптимуму [1].

Пошук новизни представляє собою підхід, який може бути інтегрований в алгоритм NEAT, щоб направляти еволюційний процес до знаходження оптимальних рішень там, де ландшафти функцій допасованості не можуть бути успішно пройдені за допомогою процесу оптимізації, заснованого виключно

на вимірюванні близькості поточного рішення до цілі. Пошук новизни має на меті усунути обмеження традиційної еволюції на основі цільової функції, заохочуючи дослідження нових областей простору рішень на основі новизни рішень, а не лише їх значення цільової функції [55].

У звичайній еволюції на основі цільової функції особини в популяції зазвичай оцінюються на основі їхньої продуктивності у виконанні конкретного завдання, і еволюційний процес надає перевагу особам, які мають успіх у цьому завданні. Незважаючи на ефективність, цей підхід може призвести до збіжності до локальних оптимумів, де рішення можуть застрягнути в неоптимальних областях простору рішень і не вдасться дослідити альтернативні рішення.

Пошук новизни відкриває іншу перспективу. Замість того, щоб зосереджуватися виключно на функції допасованості, він оцінює особини на основі їх новизни, яка кількісно визначає, наскільки поведінка або характеристики особини відрізняються від інших. Новизна визначається віддаленістю або несхожістю у поведінці чи просторі ознак.

Інтеграція пошуку новизни у алгоритм NEAT передбачає кілька ключових кроків:

1. Визначення поведінки або особливостей. Поведінка або особливості окремих особин повинні бути визначені таким чином, щоб їх можна було кількісно оцінити та порівняти. Для робототехнічних завдань це може включати визначення конкретних аспектів поведінки, таких як траєкторії, стани або сенсорні моделі.

2. Розрахунок новизни. Новизна особини визначається шляхом порівняння його поведінки чи особливостей з іншими індивідами в популяції. Міра відстані або несхожості, така як евклідова відстань або метрика подібності, використовується для обчислення показників новизни.

3. Архів. «Архів новинок» підтримується для зберігання різноманітного набору поведінки або функцій, які зустрічалися раніше. Архів служить еталоном для розрахунку новизни і допомагає запобігти передчасній збіжності популяції.

4. Відбір. Замість того, щоб відбирати особини виключно на основі функції допасованості, алгоритм NEAT із використанням пошуку новизни враховує оцінки функції допасованості та новизни. Особини з високими показниками новизни отримують шанс вижити та розмножуватися, навіть якщо їхні показники функції допасованості нижчі. Це спонукає до дослідження нових і недосліджених регіонів простору рішень.

Пошук новизни сприяє дослідженню, заохочуючи популяцію знаходити різноманітні рішення, які спочатку можуть бути пропущені традиційним відбором на основі функції допасованості.

Крім того, завдяки пріоритетності новизни алгоритм має меншу ймовірність застрягти в локальних оптимумах. Це запобігає передчасному переходу до неоптимальних рішень популяції.

Різноманітність рішень, виявлених за допомогою пошуку новизни, може призвести до більш надійних та адаптованих рішень, оскільки алгоритм розглядає більший простір пошуку.

Пошук новизни застосовувався в задачах, де дослідження різноманітних рішень має вирішальне значення. Включаючи новизну як критерій відбору, NEAT з пошуком новизни покращує здатність алгоритму знаходити інноваційні та ефективні рішення, сприяючи розвитку більш універсальних і потужних архітектур нейронних мереж [1].

3.6 Програмний засіб для адаптивного навчання нейромережевого засобу управління агентом-роборукою

Програмний засіб, що реалізовує запропоновані методи нейроеволюційного навчання моделі у програмних засобах управління агентом-роборукою на наборі середовищ, виконано мовою програмування Python. Код програмного засобу наведено у додатку Б.

Для алгоритму NEAT була обрана бібліотека NEAT-Python. Вона забезпечує реалізацію стандартних методів NEAT для моделювання генетичної еволюції геномів організмів у популяції. Зокрема, містить утиліти для

перетворення генотипу організму в його фенотип (штучну нейронну мережу) і надає зручні методи для завантаження та збереження конфігурацій геному разом із параметрами NEAT. Крім того, вона надає корисні процедури, які допомагають збирати статистичні дані про хід еволюційного процесу та зберігати/завантажувати проміжні контрольні точки. Контрольні точки дозволяють нам періодично зберігати стан еволюційного процесу і пізніше відновлювати виконання процесу [55].

Програмний засіб складається з наступних окремих компонентів (діаграма компонентів наведена на рисунку 15):

1. neat-evolve – застосунок, що здійснює роботу з параметрами нейроеволюційного процесу, завантажуючи відповідні аргументи для налаштування, а також запуск та збереження моделі нейромережі-переможця.
2. neat-test – застосунок, що здійснює оцінку точності збереженої моделі нейромережі на заданому наборі експериментів, який описується вхідними параметрами.
3. tools – пакет, що містить реалізації класів конфігурацій алгоритмів NEAT, HyperNEAT та EsHyperNEAT, а також популяцій для реалізації алгоритму NEAT з пошуком новизни та без. Крім того, пакет містить клас для реалізації логування процесу навчання та побудови графіків.
4. neat – реалізація алгоритму NEAT з допомогою бібліотеки NEAT-Python.
5. neat_gym – застосунок, що реалізує функції для пошуку новизни у алгоритмі NEAT, які використовуються у реалізації популяції з пошуком новизни NoveltyPopulation.
6. Gym – пакет, що містить реалізації набору середовищ OpenAI Gym для двовимірної задачі та тривимірної задачі з камерою, які були модифіковані для використання у розробленому програмному засобі.

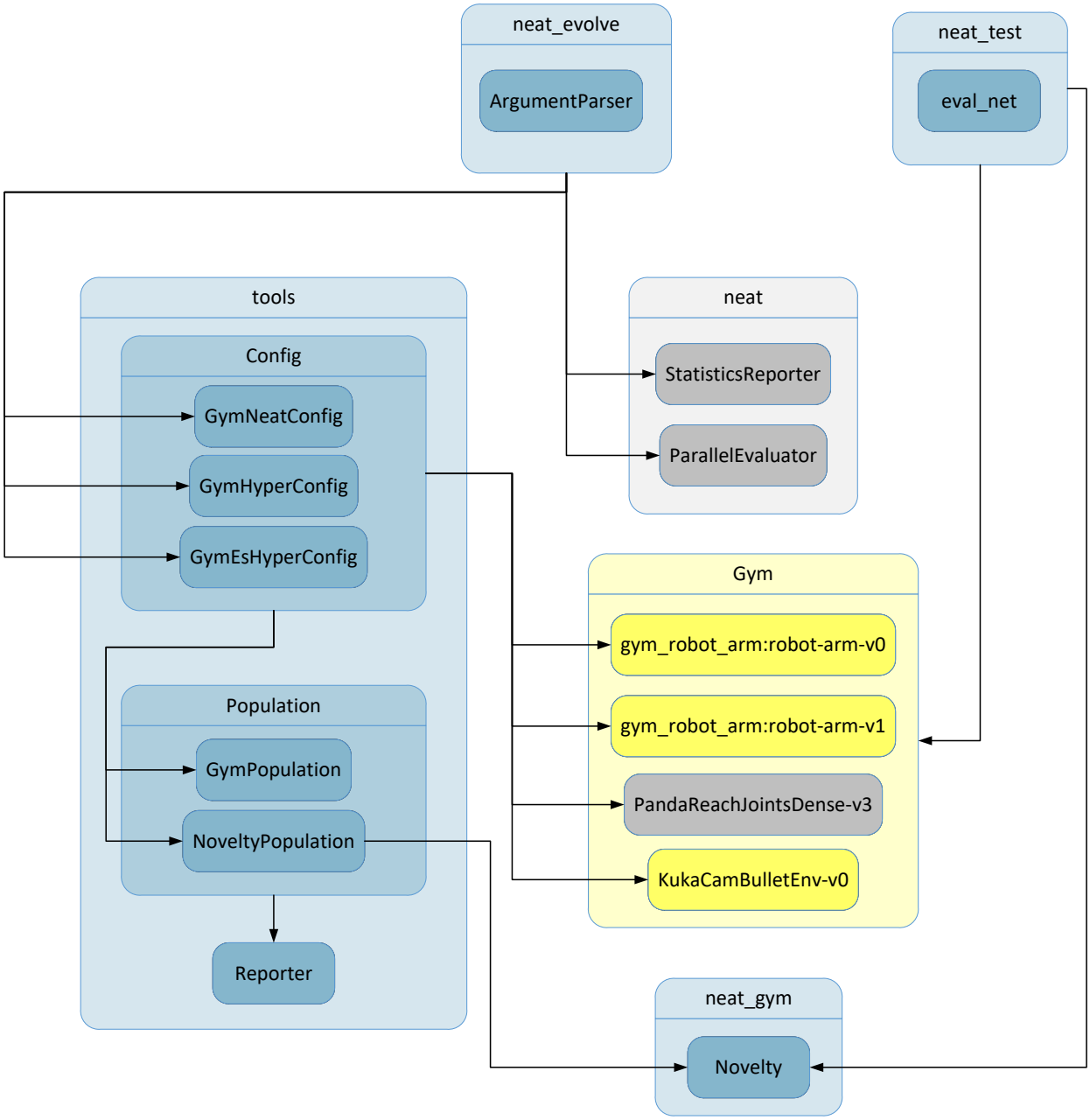


Рис. 15 Діаграма класів проєкту NEAT-Gym

Взаємодія користувача з розробленим програмним забезпеченням відбувається за наступним алгоритмом:

1. Встановити параметри навчання, конфігурацію NEAT та ідентифікатор навчального середовища у конфігураційному файлі.
2. Запустити на виконання функцію main модуля neat-evolve.
3. Завантажити файли з результатами: модель, схему структури нейромережі-переможця, графік видоутворення еволюційного процесу.
4. Запустити на виконання функцію main модуля neat-test.

5. Отримати точність натренованої моделі з консольного повідомлення.

Діаграма класів основних компонентів neat-evolve та tools наведена на рисунку 16.

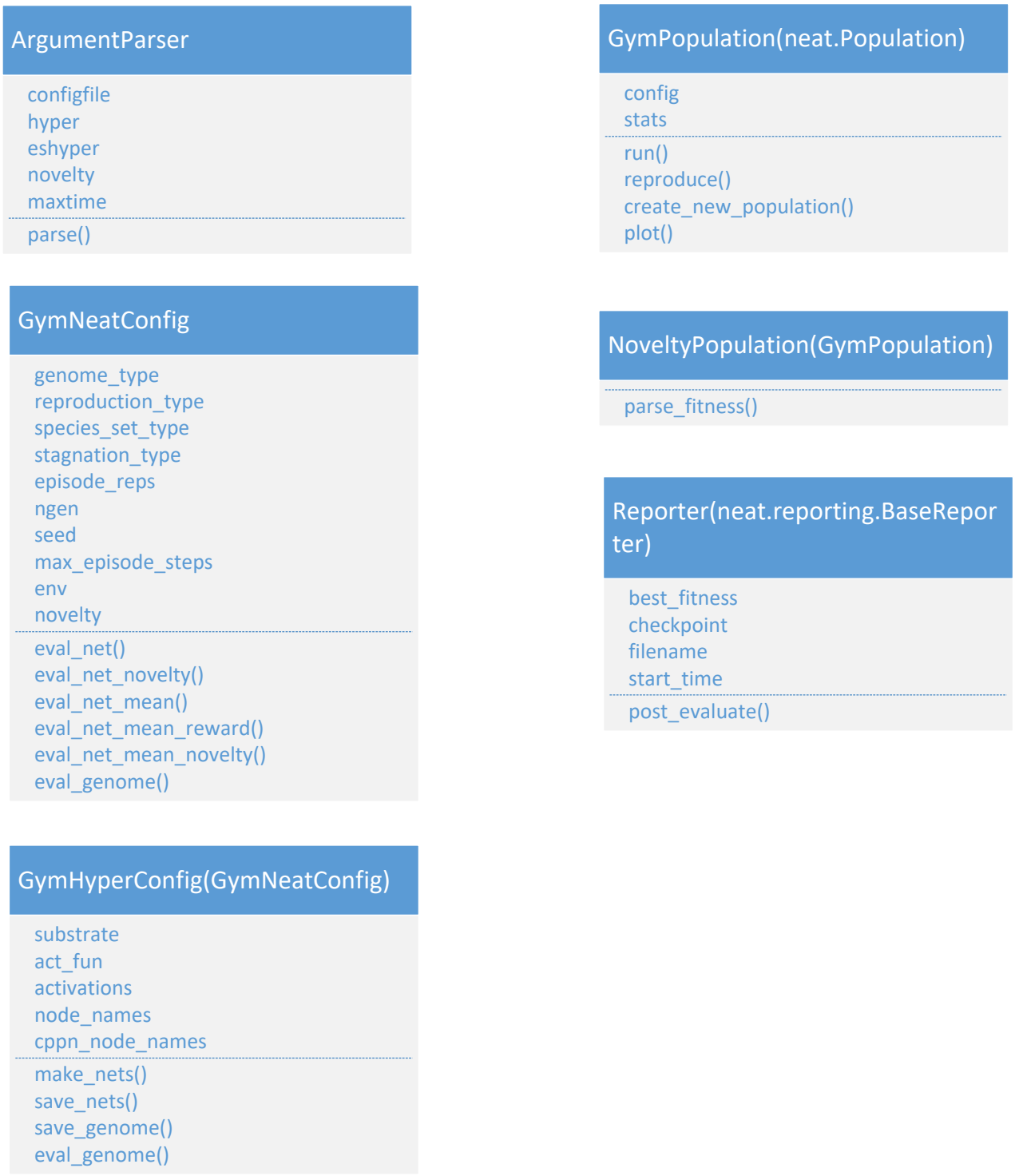


Рис. 16 Діаграма класів пакетів neat-evolve та tools

Набір середовищ може бути використаний для дослідження завдань роботизованої руки за допомогою наступних наявних інтерфейсів.

Двовимірний симулятор з дискретним простором дій. Робот складається з двох з’єднань, кожне довжиною 100 пікселів, і мета — досягти червоної крапки, яка випадковим чином генерується в кожному епізоді. Простір дій складається з керування кутами з’єднання маніпулятора робота (7 можливих управляючих подій). Простір спостережень дає інформацію про поточний стан навколишнього середовища, він є безперервним і містить відповідну інформацію про цільове та потоне положення з’єднань (4 можливі стани).

Двовимірний симулятор з неперервним простором дій. Подібно до попереднього агента, робот складається з двох з’єднань, кожне довжиною 100 пікселів, і мета — досягти червоної крапки, яка випадковим чином генерується в кожному епізоді. Але простір дій складається з неперервного сигналу керування кутами з’єднання маніпулятора робота (два значення для з’єднань у проміжку -1..1). Простір спостережень є безперервним і містить відповідну інформацію про цільове та потоне положення з’єднань (4 можливі стани).

У процесі нейроеволюції використовується одне з середовищ з симулятором роботизованої руки для реалізації методу навчання методом проб і помилок. Застосунок підтримує популяцію геномів, які розвиваються з покоління в покоління, доки не буде знайдено успішний роз’вязок. У процесі еволюції кожен організм у популяції перевіряється на придатність шляхом імітації руки з двома суглобами. Наприкінці симуляції організм отримує сигнал винагороди у вигляді кількості часових кроків, протягом яких він міг досягти цільової точки та максимальної винагороди. Отриманий сигнал винагороди визначає допасованість організму і вирішує його долю в процесі нейроеволюції.

Для прискорення процесу навчання використано розпаралелювання обчислень функції допасованості між процесорами за допомогою класу Parallel-Evaluator:

```
neat.ParallelEvaluator(mp.cpu_count(), config.eval_genome, timeout=20)
```

Параметри обраного середовища та максимальну кількість повторень у підході встановлюються у конфігураційному файлі, який обробляється модулем Config:

```
environment = gym_robot_arm:robot-arm-v1
episode_reps = 10
```

Оптимальний набір параметрів відповідної модифікації алгоритму NEAT також встановлений за допомогою конфігураційного файлу:

```
[NEAT]
compatibility_disjoint_coefficient = 1.1
[DefaultSpeciesSet]
compatibility_threshold = 3.0

[DefaultGenome]
conn_add_prob           = 0.3
conn_delete_prob        = 0.2

[DefaultStagnation]
species_fitness_func = max
max_stagnation       = 20
species_elitism      = 1

[DefaultReproduction]
survival_threshold = 0.2
min_species_size   = 2
```

Після цього обрана функція оцінки може бути встановлена та запущений процес навчання на кількості поколінь, заданій користувачем:

```
winner = population.run (parallel_evaluator.evaluate, params.ngen, params.maxtime)
```

3.7 Оцінювання ефективності пошуку новизни для двовимірної задачі досягнення цільового положення агентом-роборукою

Спершу розглянемо реалізацію навчання з використанням алгоритму NEAT без пошуку новизни. Управління еволюційним процесом відбувається з використанням цілеорієнтованої цільової функції (goal-oriented objective function). Для розглянутого агента функція допасованості приймає значення [-10, 10] залежно від успішності досягнення цільової позиції.

Досягнення цільової точки робочої кінцівки має локальні пастки найкращої допасованості у областях, де значення функції буде коливатися у проміжку [-10, 10], проте глобального наближення до кінцевої позиції не буде досягнуто.

Вибір гіперпараметрів здійснено з огляду на широку область пошуку, адже маємо велику керуючих сигналів для робочої кінцівки. Початкова конфігурація фенотипу нейромережі включає 4 вхідних вузли, 7 вихідних вузлів та 1 прихований вузол. Вузли входу відповідають параметрам, що описують поточний стан системи, а вузли виходу відповідають керуючим сигналам. Прихований вузол призначений для введення нелінійності з самого початку еволюційного процесу. Структура нейромережі представлена на рисунку 17 [1]:

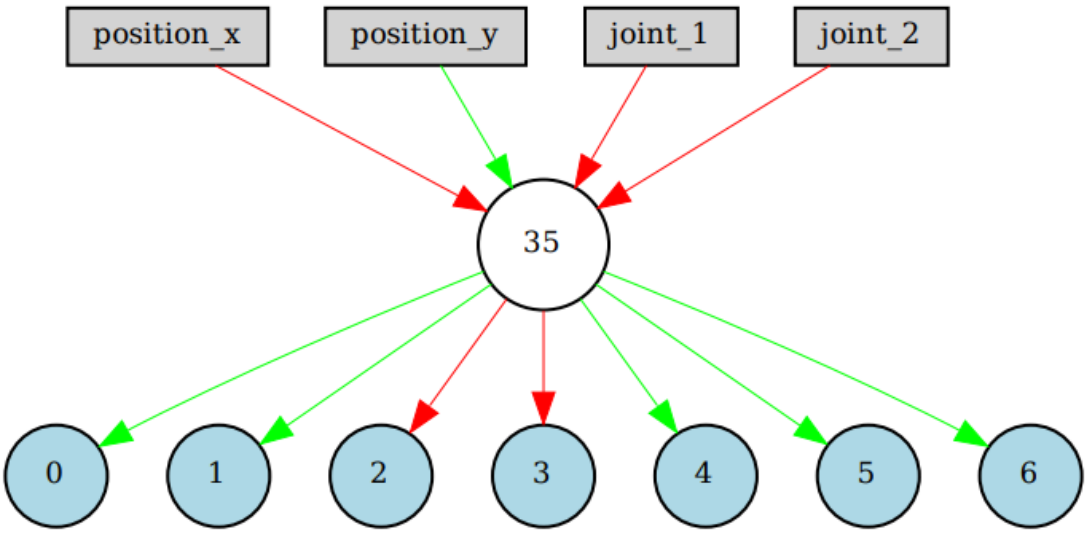


Рис. 17. Структура початкової нейромережі

Для розширення області пошуку необхідно покрити велику видову різноманітність популяції, щоб спробувати різні конфігурації геному протягом обмеженої кількості поколінь. Цього можна досягти або зниженням порогу сумісності, або збільшенням значень коефіцієнтів, що використовуються для розрахунку показників сумісності геному. Використані наступні значення параметрів:

```
[NEAT]
compatibility_disjoint_coefficient = 1.1
[DefaultSpeciesSet]
compatibility_threshold = 3.0
```

Оскільки початкова структура нейромережі має досить велику кількість виходів на невелику кількість входів, існує ймовірність перенавчання мережі, коли певні вхідні вузли не використовуватимуться і це буде приводити до субоптимального рішення, яке знаходиться в локальному мінімумі. При цьому на меті є створення оптимальної конфігурації нейромережі, яка матиме мінімальну кількість прихованих вузлів та зв'язків. Тому вірогідності додавання та видалення вузлів мають бути невеликими. Причому враховуючи початкову повнозв'язну модель, вірогідність видалення має бути додатково зменшена:

```
[DefaultGenome]
conn_add_prob           = 0.3
conn_delete_prob        = 0.2
```

За результатами експерименту навіть після 1000 поколінь еволюції успішний контролер не був знайдений. Графік видоутворення показав, що процес зійшовся до одного виду, який далі не зміг отримати ефективне рішення. Тому поріг стагнації зменшено, а кількість видів збільшено, щоб еволюційний процес підтримувався за рахунок видів, які не показали покращення значення допасованості і підлягають вимиранню:

```
[DefaultStagnation]
species_fitness_func = max
```

```
max_stagnation       = 20
species_elitism      = 1
[DefaultReproduction]
survival_threshold = 0.2
min_species_size   = 2
```

Найкращий геном, отриманий з використанням алгоритму нейроеволюції, кодує непрактичну конфігурацію нейроконтролера, яка враховує лише входи 3 та 4, що відповідають за переміщення другої ланки. Такий результат показує, що алгоритм потрапив у пастку локально оптимального рішення, коли поворот ланки, найближчої до цільової точки, може дати локальний приріст цільової функції. Причому перша ланка не була використана, хоча її поворот дозволить наблизитися до кінцевої точки.

Найкраща допасованість організму, яка була отримана, становить всього 2, що недостатньо порівняно з цільовим значенням 10 (критерієм закінчення). Структура нейромережі, що закодована цим організмом, представлена на рисунку 18.

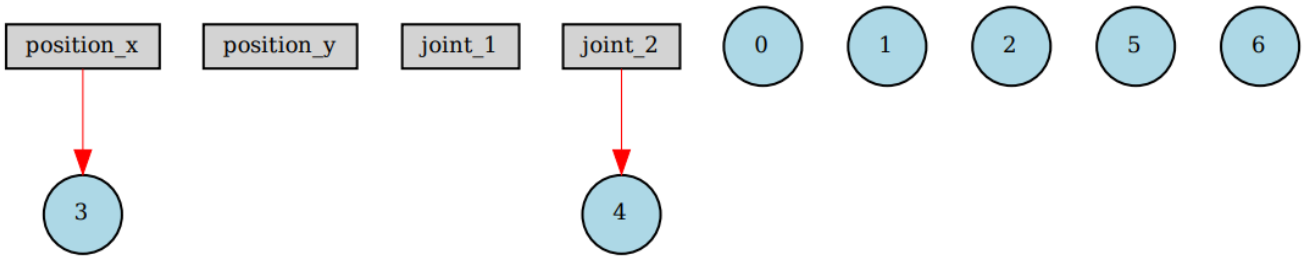


Рис. 18 Структура нейромережі, що закодована в найкращому геномі з використанням NEAT без пошуку новизни після 1000 поколінь

На графі видно, що повороти другого з’єднання залежать лише від положення по одній осі та його поточного положення. Така структура призводить до спрощених рухів, декілька з яких направляють кінцівку у напрямку цільової точки, але вони є недостатніми для її досягнення.

Графік видоутворення, представлений на рисунку 19, показує, як протягом поколінь популяції організмів розвивався процес видоутворення. Аналіз

середніх показників допасованості показав, що нейроеволюційний процес покращував показники допасованості на перших поколіннях, проте з часом він зупинився на плато, не показуючи покращень.

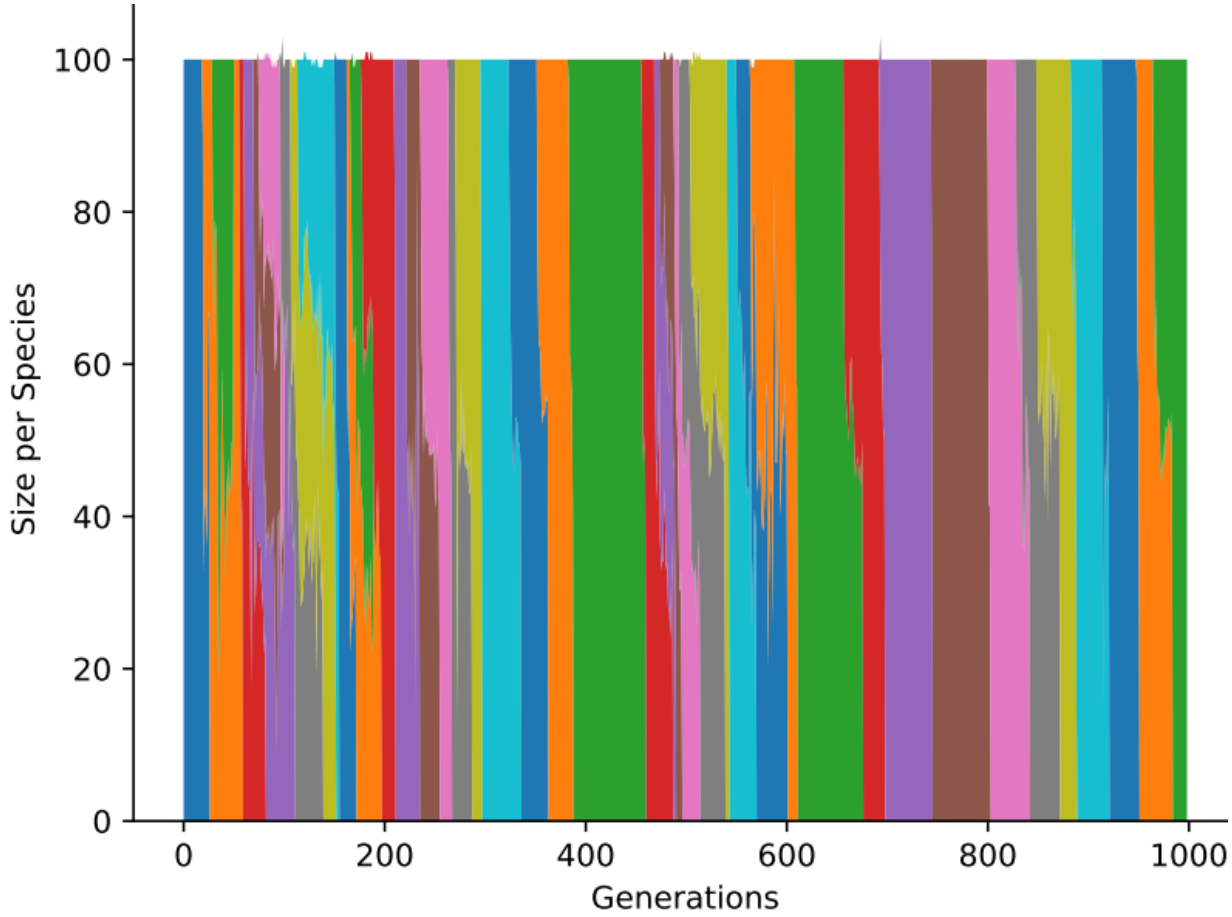


Рис. 19. Графік видоутворення еволюційного процесу з використанням NEAT без пошуку новизни

Експеримент підкреслив важливість підтримки збалансованої популяції із помірною кількістю видів. Занадто велика різноманітність видів у популяції може перешкоджати процесу нейроеволюції, зменшуючи ймовірність спарювання між організмами, що належать до різних видів. Крім того, враховуючи, що чисельність популяції фіксована, чим більше видів у популяції, тим менша чисельність кожного виду. Менші види зменшують ймовірність корисних мутацій. З іншого боку, наявність окремих видів дозволяє нам підтримувати корисні мутації в кожній ніші видоутворення та використовувати кожну мутацію в наступних поколіннях. Таким чином, занадто низька видова різноманітність також шкодить еволюції [4].

Використання нейроеволюції в пошуку політики для позиціонування роботизованої руки має враховувати компроміс між трьома параметрами системи, що розробляється:

- Обсяг тренувальних спроб, необхідних для виконання завдання за допомогою робота
- Час на виконання завдання після отримання команди управління
- Якість результату виконаного завдання (точність позиціонування) [4]

Отже, збільшення кількості поколінь не буде рішенням, необхідно використовувати механізми для покращення характеристик нейроеволюційного процесу. Тому введемо використання методу пошуку новизни, описаного у розділі 3.5.

Запуск здійснимо з тими самим гіперпараметрами на тій же кількості поколінь. Представлене середовище не має обмеження на кількість кроків, тому експеримент зависає на одному випробуванні. Виправлення вказаної проблеми потребує модифікації середовища шляхом додавання додаткової перевірки на максимально допустиму кількість кроків.

Але лише цієї зміни виявилося недостатньо, адже при оцінці запусків, які зависли у нескінченному прогоні, вони отримують максимальну оцінку і виходять з циклу з максимальним значенням функції допасованості. Для вирішення вказаної проблеми додано штраф у випадку виходу з циклу за досягнення максимальної кількості кроків, не досягнення цілі.

Структура нейромережі-переможця з додаванням пошуку новизни і модифікацій середовища представлена на рисунку 20. Як бачимо, нейромережа має складнішу топологію з більшою кількістю прихованих вузлів, але після 100 поколінь процес нейроеволюції потрапляє у локальний оптимум і зависає на неоптимальній структурі, що досягає значення функції допасованості лише -2. Причиною такої поведінки є мало видів, через занадто високий поріг max_stagnation.

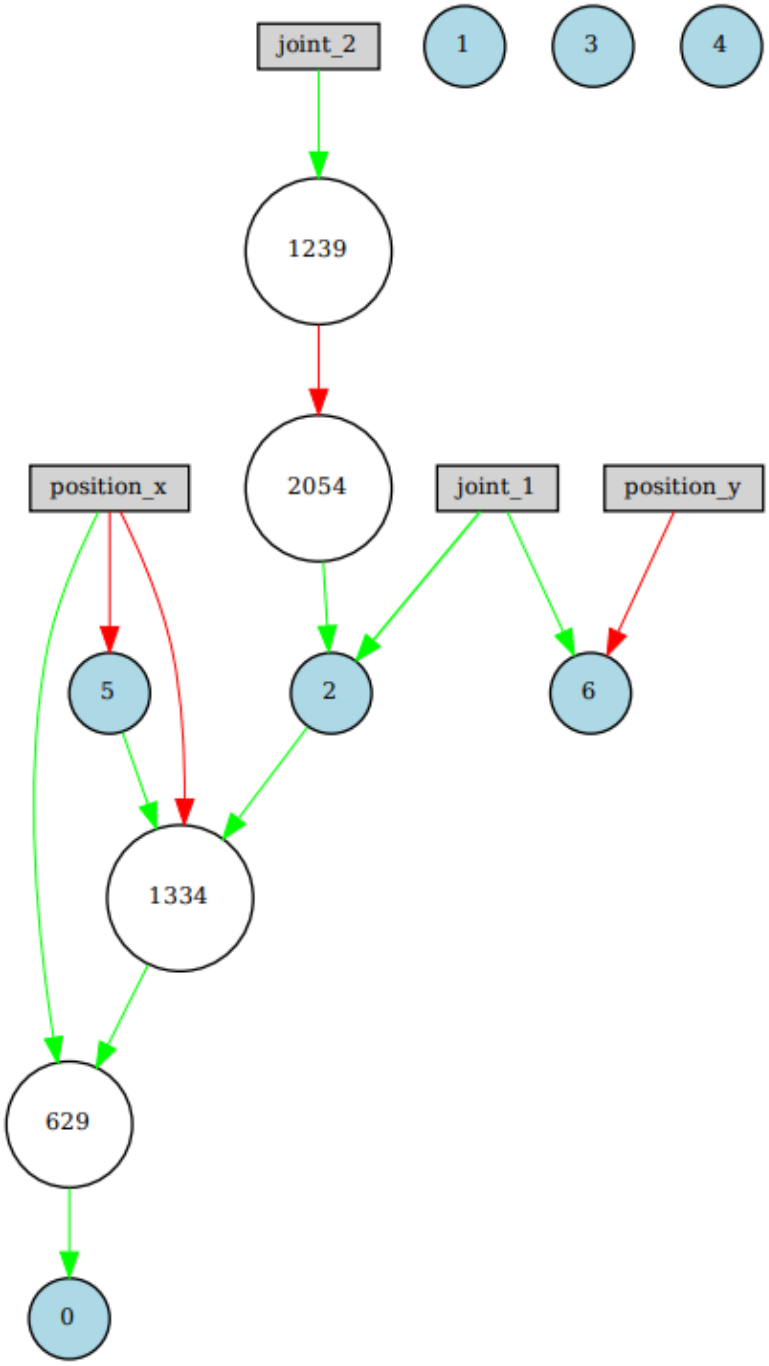


Рис. 20 Структура нейромережі-переможця з додаванням пошуку новизни і модифікацій середовища

Для вирішення вищевказаної проблеми змінюємо наступні параметри:

```
[DefaultStagnation]
max_stagnation        = 100
[DefaultReproduction]
min_species_size    = 2
```

Після 1000 поколінь еволюції найкращий агент-контролер досяг значення функції допасованості 6, що значно перевищує попередній результат. Крім того, аналіз конфігурації нейромережі найкращої особини (представлена на рисунку 21) показує, що дана мережа більш надійна, адже має зв'язки від всіх вхідних вузлів.

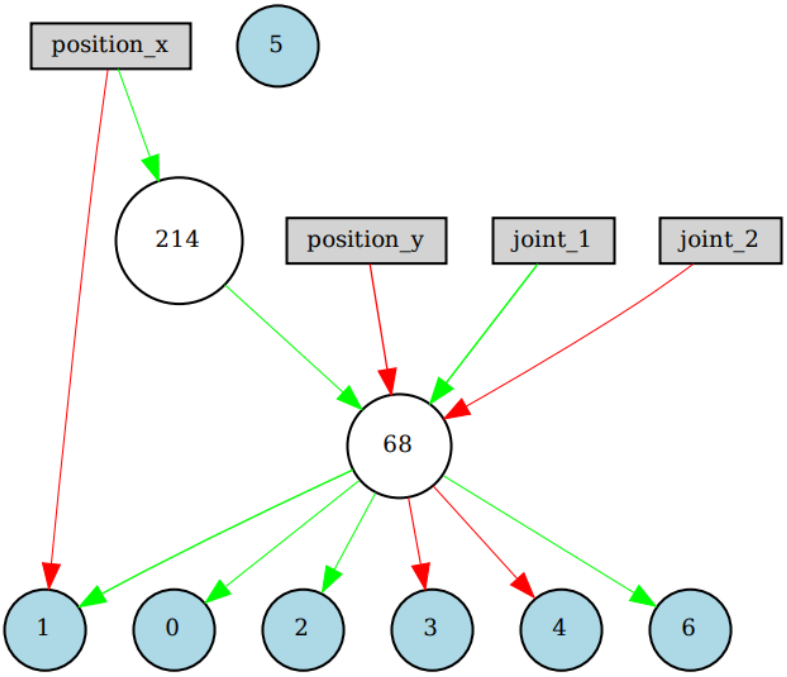


Рис. 21 Структура нейромережі, що закодована в найкращому геномі з використанням NEAT з пошуком новизни

Графік видоутворення еволюційного процесу з використанням NEAT з пошуком новизни (рисунок 22) підтверджує ефективність еволюційного процесу порівняно з аналогічним графіком без пошуку новизни. На графіку видоутворення без пошуку новизни спостерігається менша кількість видів і відповідне домінування кількох видів, що не гарантує еволюцію корисних стратегій виду. І, як наслідок, ефективність еволюції найкращого геному, який успішно розв'язуватиме поставлену задачу досягнення цільової точки, є низькою. На відміну від графіку видоутворення з пошуком новизни, де види розвиваються більшу кількість поколінь, таким чином зберігаючи корисні властивості, які можуть бути використані майбутніми видами у наступних поколіннях.

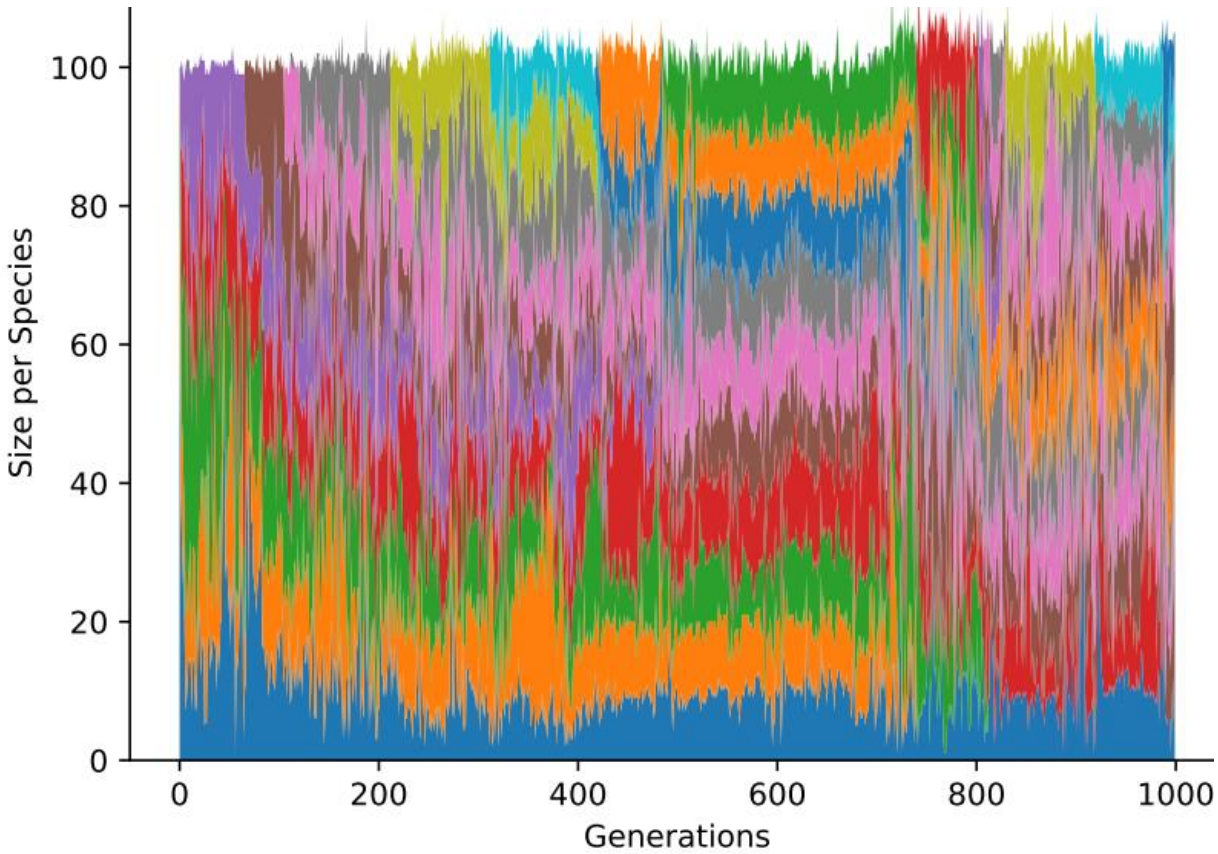


Рис. 22 Графік видоутворення еволюційного процесу з використанням NEAT з пошуком новизни

Збільшення кількості поколінь дозволить мережі дорозвинути переможця для отримання ефективного контролеру з мінімальною конфігурацією та керуванням всіма доступними управляючими сигналами. Зокрема одночасним приростом кутів з'єднань 1 і 2 (вихідний вузол 5), який не має розвинутих зв'язків у отриманій конфігурації з найкращого геному [1].

3.7 Висновки до третього розділу

В даному розділі розглянуто можливість використання нейроеволюційного підходу для реалізації адаптивного навчання моделі у програмних засобах управління роборукою. Розглянутий підхід показує перспективність застосування нейроеволюції для автоматизації розробки нейронних мереж для задач управління роботизованими системами з великими просторами рішень.

Використання нейроеволюції в управлінні рукою робота надає кілька переваг. Вона може обробляти багатовимірні простори стану та дії, забезпечуючи гнучкість для керування складними роботами. Вона також здатна виявляти політики керування, які демонструють непередбачувану поведінку, явно не запрограмовану інженерами. Нейроеволюція дозволяє паралельно оцінювати кілька нейронних мереж, забезпечуючи ефективне дослідження простору пошуку.

Використовуючи переваги NEAT, можливо використовувати еволюційні процеси для розвитку нейронних мереж, здатних до точного та адаптивного керування позиціонуванням роботизованої руки. Гнучкі топології, можливості оптимізації та здатність обробляти складну динаміку сприяють підвищенню продуктивності, адаптивності та ефективності систем позиціонування роботизованої руки.

Однак нейроеволюція також стикається з проблемами. Вона може бути обчислювально затратною, особливо при роботі з великими нейронними мережами та складними завданнями.

Розглянуто реалізацію процесу навчання політики в формі нейромережі для управління двовимірною роборукою з двома ланками. Згідно результатів експериментів відмічається підвищення ефективності найкращого рішення, знайденого із застосуванням пошуку новизни для алгоритму NEAT порівняно з алгоритмом NEAT без пошуку новизни. Встановлено, що запропонований підхід дозволяє отримати ефективну політику в формі нейромережі, яка має мінімальну конфігурацію, що дозволяє підвищити швидкодію контролера, яка є критичною для роботи реальної системи.

Таким чином, використання пошуку новизни як методу оптимізації нейроеволюційного процесу для вирішення задачі позиціонування дозволяє підвищити ефективність навчального процесу та отримати оптимальну топологію мережі.

Розроблена реалізація системи для навчання мережі програмного управління роботизованим агентом з робочою кінцівкою використана в

наступному розділі для розроблення програмного засобу навчання моделі у програмних засобах управління роботизованим агентом, який отримує інформацію про середовище з зображень камери, тобто у більш наближений до реальних умов спосіб.

РОЗДІЛ 4. МЕТОД АДАПТИВНОГО НАВЧАННЯ МОДЕЛІ У ПРОГРАМНИХ ЗАСОБАХ УПРАВЛІННЯ РОБОРУКОЮ З ВИКОРИСТАННЯМ ДАНИХ З КАМЕРИ

4.1 Метод HyperNEAT для системи з вхідними даними у форматі зображень

Алгоритм NEAT є ефективним для задачі управління роботизованою рукою, де вхідні дані представлені набором даних з кількох датчиків, наприклад, двома координатами кожного з'єднання. Якщо ж вхідна інформація про стан агента надходить з камери, що розміщена на робочій кінцівці або на основі робота, задача стає на порядок складнішою. Для такої системи доцільним є використання алгоритму HyperNEAT [53].

HyperNEAT (Hypercube-based Neuro-Evolution of Augmenting Topologies) є розширенням NEAT, яке використовує форму непрямого кодування під назвою композиційні шаблони створення мереж (CPPN). Воно було створено як спроба скоротити розрив між результатами, отриманими алгоритмами нейроеволюції, та масштабом природного мозку.

У CPPN вхідні дані є декартовими координатами в n-вимірному просторі, де n є кількістю входів, і передаються у функцію f. Значення f, оцінене за кожною координатою, забезпечує наявність, відсутність або інтенсивність вираження точки в n-вимірному просторі, і тому фенотип, отриманий оцінкою f (який можна розглядати як генотип), є шаблоном. Різниця між звичайними штучними нейронними мережами і CPPN полягає в тому, що штучні нейронні мережі зазвичай містять лише сигмоїдальні функції, тоді як CPPN можуть містити багато різних типів функцій.

Хоча CPPN можна розробити за допомогою генетичних алгоритмів так само, як і штучні нейронні мережі, у HyperNEAT CPPN не розробляються як кінцевий результат, а використовуються для кодування нейромережі. Оскільки CPPN створює просторовий шаблон, а нейромережа є шаблоном зв'язків, алгоритм потребує відображення між просторовим шаблоном і шаблоном

зв’язку. Вхідними даними в CPPN є дві точки, а не одна, і дві точки представляють два вузли в нейромережі, а вихід функції CPPN є вагою зв’язку між двома вузлами. Таким чином, модель, створена CPPN, є топологією нейронної мережі. Цей CPPN, що створює топологію, називається з’єднувальним CPPN, на відміну від просторових CPPN, які створюють просторові моделі. З’єднувальний CPPN бере сітку вузлів, яка називається субстратом, і запитує кожне потенційне з’єднання, щоб визначити, чи дійсно існує з’єднання та яка його вага, беручи положення двох вузлів і знаходить вагу з’єднання між ними. Це створює шаблон з’єднань між вузлами в просторі, який є похідним від конфігурації субстрату.

HyperNEAT розвиває CPPN за допомогою стратегії NEAT. CPPN є представленнями шаблонів у гіперпросторі, де кожна точка обмежена чотиривимірним гіперкубом. Отже, шаблон у чотиривимірному гіперкубі можна відобразити на двовимірній нейронній мережі. Першим кроком HyperNEAT є вибір субстрату та зв'язок між входами та виходами. Алгоритм ініціалізує перше покоління мінімальних CPPN з випадковими вагами, так само як NEAT ініціалізує перше покоління мінімальних штучних нейронних мереж з випадковими вагами. Потім він запитує кожну CPPN, і створює шаблон зв'язку, подібний до нейромережі. Алгоритм визначає придатність цієї нейромережі так само, як це робить NEAT зі своїми мережами, і розвиває друге покоління CPPN відповідно до стратегії NEAT. Це повторюється, як і будь-який алгоритм нейроеволюції, доки не буде знайдено рішення.

HyperNEAT особливо добре підходить для задач, де важлива інформація про геометричні представлення. За допомогою субстрату він «бачить» координати нейронів. Він може «бачити», який піксель поруч із яким на зображенні або який квадрат біля якого на шахівниці, замість того, щоб вивчати їх на прикладах, як це робила б традиційна нейронна мережа [2].

Зауважимо, що HyperNEAT не обов’язково кращим вибором ніж традиційний NEAT для вирішення будь-якої задачі. Хоча HyperNEAT повинен працювати принаймні так само добре як NEAT у більшості задач, можливо, не

завжди варто витрачати ресурси розробника щодо встановлення субстрату, якщо це не дасть значної переваги. Тому розглянемо випадки, коли HyperNEAT забезпечує підвищення ефективності.

Задачі з великою кількістю входів. Оскільки CPPN у HyperNEAT є непрямим кодуванням, розмірність мережі не є основним фактором складності проблеми для HyperNEAT, на відміну від більшості інших підходів навчання. Це означає, що великі масиви вхідних даних не є проблемою для алгоритму навчання, хоча ЦП може потребувати більше часу, щоб фактично пройти по дуже великій мережі. Деякими прикладами задач із великими масивами вхідних даних є предметна область з використанням комп'ютерного зору та настільні ігри.

Задачі з великою кількістю виходів. На відміну від типових алгоритмів навчання з підкріпленням, HyperNEAT так само добре працює з багатьма виходами, як і з багатьма вхідними даними. Традиційні нейронні мережі зазвичай не містять десятків, сотень або більше виходів. Крім того, наявність великої кількості виходів може додати творчий підхід до вирішення задачі. Наприклад, субстрат може виводити цілу двовимірну карту, на якій зображено, куди агенту потрібно рухатися.

Задачі зі змінною роздільною здатністю. Для деяких задач може бути незрозуміло, скільки саме має бути входів. Наприклад, оптимальна роздільна здатність поля зору може бути невідома апріорі. Подібним чином найкраща роздільна здатність для виходів також може бути невідома. Наприклад, оптимальна кількість сегментів і з'єднань у багатосегментній руці може бути невідома, а кожен з них потребує окремого виходу. HyperNEAT є гарним вибором для таких предметних областей, оскільки його ефективність значною мірою не залежить від наданої роздільної здатності. Тому розробнику не потрібно турбуватися про те, щоб спочатку отримати правильну роздільну здатність або навіть мінімізувати її. Якщо пізніше потрібно буде отримати більшу роздільну здатність, можна повторно запустити експеримент із вищою роздільною здатністю з високим ступенем упевненості, що він все одно

працюватиме. Крім того, іноді можливо просто збільшити роздільну здатність субстрату після навчання, без будь-якого додаткового навчання, і повторно запустити CPPN через субстрат. Іноді рішення все одно працюватиме з вищою роздільною здатністю, навіть якщо воно не було навчено на цій роздільній здатності. Навіть якщо це не так, можливо продовжити тренування на вищій роздільній здатності та таким чином почати донавчатися на основі того, що було вивчено на нижчій роздільній здатності.

Задачі з геометричними співвідношеннями між входами та/або виходами. HyperNEAT бачить координати вхідних нейронів, вихідних нейронів і прихованих нейронів. Це означає, що він може використовувати переваги їхньої відносної геометрії для вирішення проблем. Наприклад, у шашках він фактично бачить, яке поле прилягає до якого поля та в якому напрямку. Такі знання є основоположними для такої гри, як шашки, а це означає, що геометрія субстрату стає значною перевагою порівняно, наприклад, із традиційними нейронними мережами, які взагалі не мають геометричних знань, а натомість вони повинні вивчити це через багато прикладів, що погіршує кінцеву ефективність. Але геометрія важлива не лише в настільних іграх. Те, який піксель поруч із яким, має значення також на зображенні. Крім того, має значення який суглоб поруч із яким у тілі, що відобразиться на геометрії виходів. Тобто, геометричні кореляції між входами та виходами також можуть бути корисними. Якщо лівий датчик знаходиться ліворуч, а ліва робоча кінцівка — ліворуч (а права є відображенням лівого), HyperNEAT може використовувати симетрію геометрії та співвідносити ліве з лівим і праве з правим. Можливість передавати геометричні знання HyperNEAT через субстрат створює нову можливість досліджувати геометричні представлення предметних областей роботизованих систем.

Багатоагентні задачі навчання. Розширення HyperNEAT під назвою мультиагентний HyperNEAT дозволяє застосувати потужність непрямого кодування та геометрії до багатоагентних доменів навчання. Причина, по якій HyperNEAT підходить для такого типу задач, полягає в тому, що агент-команда

часто має неявну геометрію (геометрію політики), у якій геометрична позиція гравця в команді корелює з його роботою. Наприклад, у футбольній команді захисники займають задню частину поля, а нападники — передню. Ці типи кореляцій поширені в спортивних командах і можуть використовуватися також в інших типах команд. Розподіл робочих позицій у геометрії команди схожий на геометричний візерунок, де кожна робота має інший колір на позиціях команди.

Нейронна пластичність. Дослідження з адаптивним HyperNEAT показали, що CPPN можуть кодувати, як мають змінюватися ваги з'єднань у відповідь на рівні активації навколишніх нейронів. Насправді, якщо подати правило зміни ваг з'єднання як колір, то кодування набору таких правил по всій геометрії мережі схоже на малювання кольорового візерунка по її з'єднаннях. Таким чином, CPPN може кодувати геометрію правил пластичності в мережі.

4.2 Застосування методу на основі гіперкуба для адаптивного навчання роботизованого агенту

Як було визначено раніше, «Hyper» в назві метода HyperNEAT означає «кодування на основі гіперкуба». HyperNEAT базується на ключових аспектах: він призначений для використання геометричної регулярності у вхідному та/або вихідному просторі; він може розвивати дуже великі мережі, які можуть обробляти вхідні дані дуже великої розмірності [94]; і, незважаючи на те, що ідея методу походить від природних процесів розвитку, він фактично не використовує явну стадію розвитку, а натомість намагається абстрактно імітувати деякі результати природного розвитку, що робить його відносно ефективним порівняно з іншими підходами непрямого кодування.

Кодування гіперкуба має дві аспекти: це метод відображення геометрії фенотипу субстрату штучної нейронної мережі на його шаблон зв'язків на основі гіперкубу; а також метод створення вагової функції, яка виконує це відображення. Опис HyperNEAT почнемо з типів функцій, які виконують відображення, а потім опису, як вони застосовуються в схемі кодування

гіперкуба. Вагова функція, яка утворює геном, може бути будь-якою функцією, яка приймає кілька вхідних даних і дає дійсний результат. Функція може мати різноманітне представлення: від таблиці пошуку чи програми до композиції або мережі багатьох типів функцій. Останній тип функції використовується в оригінальному формулюванні HyperNEAT і є частиною, що відповідає за «NEAT» у «HyperNEAT». Як зазначено у розділі 1, NEAT відноситься до методу нейроеволюції з використанням прямого кодування, адаптованого для використання багатьох типів функцій активації, відомого як композиційна мережа формування шаблону (CPPN, рисунок 23) [95]. Така функція з двома входами може бути використана для отримання двовимірного зображення інтенсивності. Типи створюваних зображень зазвичай мають природні характеристики, такі як повторення, повторення з варіаціями та симетрія з деякими плавними та/або різкими градієнтами.

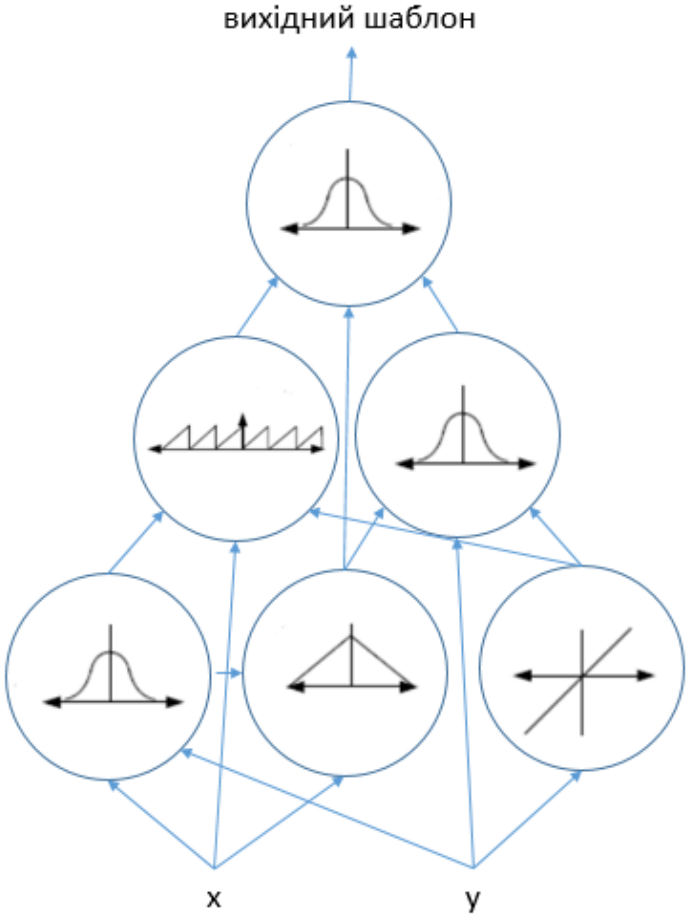


Рис. 23 Двовимірна активація CPPN

Далі це двовимірне зображення можна використовувати для визначення ваг з’єднань або шаблону ваг одного нейрона в мережі прямого поширення з шарами, що складаються з двовимірних масивів нейронів. Вхідними даними функції є координати вихідних (або цільових) нейронів, а виходом є значення ваг між вихідним (або цільовим) нейроном і відповідним нейроном.

HyperNEAT розширює представлену ідею, щоб використовувати єдину функцію для визначення ваг більш ніж одного нейрона в потенційно більш ніж одному шарі або фактично в будь-якій топології мережі в декартовому просторі. Наприклад, якщо потрібно, щоб функція вказувала вагові коефіцієнти лише між двома шарами нейронів, то знадобляться 4 вхідні дані: координати x і y як для вихідного нейрона, так і для цільового нейрона. Таким чином, шаблон ваг мережі кодується в «зображенні» гіперкуба, причому інтенсивності в різних точках представляють вагові зв’язки між різними нейронами. Це можна розширити до більш ніж двох рівнів, використовуючи ще два входи в мережу, по одному для координат вихідного та цільового шарів (або лише один додатковий вхід для мережі прямого поширення, оскільки цільовий рівень є неявним).

Для генерації фенотипу відбувається запит до вагової функції щодо значення ваг між кожною парою нейронів, які можуть мати зв’язок між собою (наприклад, топологія може бути обмежена лише прямим поширенням) шляхом введення координат вихідного та цільового нейронів, і беручи результат як значення ваги. Якщо вказане значення ваги менше порогового значення, воно не буде розглядатися. Розміри та топологію мережі необхідно вказати вручну.

Також можна використовувати топології, відмінні від сіток, і вводити геометричну інформацію, відмінну від координат, у вагову функцію, наприклад відстань або кут між нейронами. Д'Амбросіо та Стенлі [96] експериментували з використанням двох топологій завдання зі збирання їжі роботом, де він мав круглу форму з датчиками дальності, рівномірно розподіленими навколо його тіла. Одна топологія розташовувала вхідний і вихідний шари (не було

прихованих нейронів) на прямих лініях, а інша розташовувала їх концентричними кільцями, що більш природно представляло розташування датчиків. Вхідними даними для вагової функції завжди були координати нейронів. Експеримент показав, що лінійне розташування мало дещо кращі результати навчання. Автори також спробували додати відстань між вихідним і цільовим нейронами до вхідних даних вагової функції; це значно покращило час навчання та продуктивність узагальнення, а також зрівняло продуктивність двох топологій.

Вагова функція може створювати функції з повторенням, повторенням із варіацією та симетрією. Таким чином, шаблони ваг у фенотипі штучної нейронної мержі, яка створена HyperNEAT, також мають ці характеристики, що дозволяє кодувати та еволюціонувати повторювані структури та модулі.

Оскільки HyperNEAT явно кодує ваги фенотипу нейронної мережі на основі геометричних зв’язків, він здатний безпосередньо використовувати геометричні закономірності вхідного та/або вихідного простору. Наприклад, якщо вхідний простір — це набір датчиків дальності, розташованих за регулярним шаблоном навколо робота, то, розташувавши їх у такому самому порядку на вхідному рівні мережі, HyperNEAT автоматично зможе використовувати цей зв’язок і будь-які геометричні закономірності, властиві різновидам шаблонів, які з’являються у вхідних даних. Наприклад, якщо виявлення стін має відношення до завдання, кодуванню відносно легко створити шаблон ваг, який виявляє, коли кілька сусідніх датчиків реєструють щось поруч, і застосувати цей шаблон ваг для всіх датчиків. Здатність безпосередньо використовувати геометричну закономірність видається унікальною в галузі нейроеволюції та відкриває низку завдань, для вирішення яких може бути використаний представлений метод.

Як зазначено у розділі 1, NEAT базується на трьох концепціях: відстеження генетичної спадщини для визначення подібності між мережами та операторами кросинговеру; видоутворення для збереження генетичної різноманітності та захисту нових інновацій; і ускладнення шляхом додавання

структурних елементів протягом поколінь. NEAT використовується в HyperNEAT для створення вагових функцій, дозволяючи мережам містити багато типів функцій активації.

Метод NEAT використовується для відстеження всіх генів, введених під час еволюції, причому кожен ген кодує або з'єднання, або нейрон. Кожному новому гену, створеному через структурні мутації, присвоюється унікальне історичне маркування, або ідентифікатор. Це дозволяє визначити, які індивідууми з якими сумісні, і як їхні гени можна комбінувати для створення нових мереж. Ідентифікатори генів зберігаються під час кросинговеру, тому операції кросинговеру можна виконувати без дорогого топологічного аналізу мереж.

NEAT реалізує метод видоутворення, щоб допомогти зберегти генетичне різноманіття. Видова приналежність визначається кількістю ідентичних і різних генів між представником виду і потенційним членом. Використовується підхід до обміну придатністю, коли придатність особини ділиться на кількість особин у її виді. Це дозволяє індивідам конкурувати переважно в межах своїх власних ніш (видів), а не з усією популяцією, дозволяючи топологічним інноваціям мати час для оптимізації, перш ніж конкурувати з іншими нішами в популяції.

Початкова сукупність мереж не містить прихованих вузлів і відрізняється лише початковими значеннями ваг. У ході еволюції мережі поступово додають структури через мутації, а нові інновації захищаються за допомогою видоутворення.

Одним з обмежень оригінального методу HyperNEAT є те, що розмір і топологію мережі потрібно вказувати вручну. Щоб подолати це обмеження, існує розширення під назвою Evolvable Substrate HyperNEAT (ES-HyperNEAT), яке дозволяє еволюціонувати розміщення та щільність нейронів у субстраті, базуючись на припущенні, що інформація про те, де розмістити вузли, неявно закодована у шаблоні ваг. Таке розширення — це спосіб інтерпретації виходу вагової функції, щоб вирішити, куди слід розмістити

нейрони на основі складності виходу в заданому діапазоні в просторі субстрату, тобто там, де є більша складність вагової моделі, там більше нейронів. Конкретний метод, який використовується для визначення складності та конкретного місця розміщення вузлів, ефективно визначає метод, який вагова функція має вивчити, щоб розмістити вузли. Як правило, ES-HyperNEAT має такі ж результати навчання, як і HyperNEAT, хоча він також повинен був розвинути розміщення нейронів, однак він використовував набагато більше вузлів, ніж було потрібно та використано в HyperNEAT для виконання завдання.

4.3 Програмний засіб для адаптивного навчання агента-роборуки на основі вхідних зображень

Програмний засіб для адаптивного навчання нейромережі, що був розглянутий у розділі 3.6, доповнений для задачі маніпулювання не лише на основі даних з датчиків агента і цільового положення точки, але і даних з камери, яка розташована безпосередньо на агенті. Це наближує розглянуту задачу до умов реального світу, де точна позиція цільової точки як правило невідома. Програмний засіб має забезпечувати підтримку як середовища для простору дій агента-роборуки, так і можливості використання необхідного нейроеволюційного алгоритму зі зручним налаштуванням параметрів та інструментами для аналізу результатів.

Набір середовищ був доповнений двома середовищами відповідно до поставленої задачі. Обидва реалізовані на основі бібліотеки PyBullet.

Тривимірний симулятор PandaReach. Симулятор роборуки Franka Emika Panda (рисунок 24), який має на меті позиціонування робочої кінцівки у цільову точку тривимірного середовища. Значення винагороди повертається додатне, якщо цільова позиція досягнута. Управляюча дія відповідає переміщенню кінцевого елемента роборуки.

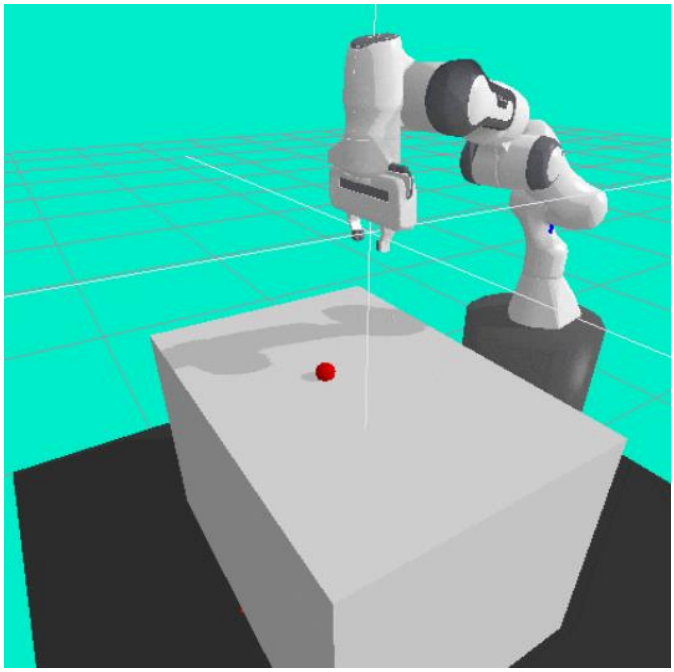


Рис. 24 Зображення з камери середовища з тривимірним симулятором PandaReach

Управління агентом здійснюється за допомогою детермінованих управляючих сигналів залежно від отриманих спостережень. Інтерфейс середовища включає набір стандартних для OpenAI Gym функцій. Приклад функції для досягнення цільової точки, коли позиція точки відома:

```
import gymnasium as gym
import panda_gym

environment = gym.make("PandaReach-v3")
observation, info = environment.reset()
for i in range(100):
    cur_pos = observation["observation"][0:3]
    des_pos = observation["desired_goal"][0:3]
    action = 5.0 * (des_pos - cur_pos)
    observation, reward, terminated, truncated, info = env.step(action)
    if terminated or truncated:
        observation, info = env.reset()
```

```
env.close()
```

Управління параметрами камери включає в себе взаємодію через інтерфейс Viewpoint. Параметри, доступні до налаштування:

- render_width (int): ширина зображення
- render_height (int): висота зображення
- render_target_position (np.ndarray): позиція камери
- render_distance (float): відстань до камери
- render_yaw (float): поворот камери навколо вертикальної вісі
- render_pitch (float): поворот камери навколо поперечної вісі
- render_roll (int): поворот навколо поздовжньої вісі

Інтерфейс налаштування камери дозволяє експериментувати зі зміною зовнішніх параметрів камери, проте є обмеженим для змін внутрішніх параметрів. Як описано у розділі 2, налаштування всіх параметрів камери дозволить наблизити умови експерименту до умов реального середовища для навчання моделі під конкретні параметри робота та камери, встановленої на ньому. Оскільки зміни до вихідного коду середовища PandaReach для додавання функціоналу зміни внутрішніх параметрів камери є неможливими, вирішено для дослідження обрати аналогічне середовище, проте з доступним кодом для модифікації під поставлені задачі.

Тривимірний симулятор KukaCamBulletEnv. Середовище, у якому спостереження представлене зображенням з камери, що розміщена на платформі-основі робота (рисунок 25). Керуючими діями є дискретні значення поворотів відповідних кутів з’єднання.

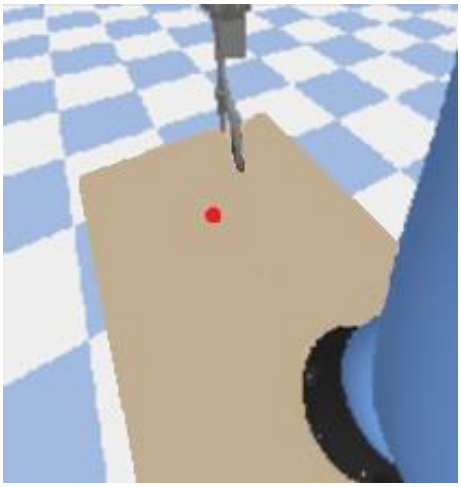


Рис. 25 Зображення з камери середовища з тривимірним симулятором KukaCamBulletEnv

Інтерфейс середовища пропонує метод `resetDebugVisualizerCamera` для встановлення положення камери аналогічно до попереднього середовища. Параметри, доступні до налаштування:

- cameraDistance(float): відстань до камери
- cameraYaw (float): поворот камери навколо вертикальної вісі
- cameraPitch (float): поворот камери навколо поперечної вісі
- cameraTargetPosition (np.ndarray): позиція камери
- physicsClientId (int): ідентифікатор

Зміни внутрішніх параметрів камери також не передбачені, проте вихідний код середовища може бути змінений для підтримки встановлення необхідних значень. Зокрема, для симуляції камери у pybullet повинні бути розраховані матриці перетворення.

Перша функція cameraMatrixToBullet перетворює матрицю внутрішніх параметрів, отриману в результаті калібровки (процес калібровки камери описаний у розділі 2), на матрицю проекції, що використовується в OpenGL і Pybullet:

```
from pyquaternion import Quaternion
import numpy as np

def cameraMatrixToBullet (K, w, h, near, far):
```

```
    f_x = K[0,0]
    f_y = K[1,1]
    c_x = K[0,2]
    c_y = K[1,2]
    A = (near + far)/(near - far)
    B = 2 * near * far / (near - far)

    projection_matrix = [
    [2/w * f_x,  0,          (w - 2*c_x)/w,  0],
    [0,          2/h * f_y,  (2*c_y - h)/h,  0],
    [0,          0,          A,              B],
    [0,          0,          -1,             0]]
    return np.array(projection_matrix).T.reshape(16).tolist()
```

Друга функція cameraPoseToBullet перетворює зовнішні параметри: поворот та позицію, на матрицю, що використовується в OpenGL і Pybullet:

```
from pyquaternion import Quaternion
import numpy as np

def cameraPoseToBullet (q, t):
    q = Quaternion([q[3], q[0], q[1], q[2]])
    R = q.rotation_matrix
    T = np.vstack([np.hstack([R, np.array(t).reshape(3,1)]),
                            np.array([0, 0, 0, 1])])
    Tc = np.array([[1,   0,    0,  0],
                   [0,  -1,    0,  0],
                   [0,   0,   -1,  0],
                   [0,   0,    0,  1]]).reshape(4,4)

    T=Tc@np.linalg.inv(T)
    viewM = T.T.reshape(16)
    return viewM
```

Зображення з камери агента може бути модифіковане з врахуванням матриць, отриманих представленими функціями:

```
projectionMatrix = cameraMatrixToBullet (K, w, h, near, far)
viewMatrix = cameraPoseToBullet(q, t)
_, _, rgb, depth, segmentation = bullet.getCameraImage(W, H, viewMatrix, projectionMatrix, shadow = True)
```

Вибір відповідної модифікації алгоритму NEAT може бути здійснений шляхом заповнення конфігураційного файлу, обробку якого здійснює модуль Config:

```
if is_hyper:
    config = GymHyperConfig(args)
if is_eshyper:
    config = GymEsHyperConfig(args)
else:
    config = GymNeatConfig(args)
```

4.4 Оцінювання ефективності методу навчання на основі гіперкуба для програмного управління в задачах роборуки на основі даних з камери

Метою оцінки ефективності нейроеволюційного методу навчання моделі у програмних засобах управління в задачах роборуки на основі даних з камери є дослідження можливостей та властивостей HyperNEAT в задачах управління агентом-роборукою. Для цього проведено експерименти із застосування HyperNEAT до різних завдань візуальної обробки з різним ступенем складності та типами регулярності та аналізом результатів.

4.4.1 Задачі позиціонування кінцівки та керування роботизованим агентом

Зображення сцени згори традиційно використовується в задачах навчання з підкріпленням для навігації робота [97]. Кілька вхідних шарів використовуються для різних елементів карти або в різних масштабах карти.

Завдання полягає в тому, щоб досліджувати місцевість, уникати перешкод і знаходити цільову позицію, яку не можна було побачити (тому основною стратегією є відвідування регіонів, які раніше не були відвідані).

Для задачі позиціонування робочої кінцівки, на відміну від задачі навігації, більш наближеним до реальних умов є використання зображень з камери, яка встановлена безпосередньо на робочій кінцівці або на основі робота. У найпростішому представленні задача може вимагати, щоб робоча кінцівка рухалася до цільового об’єкта на порожній сцені, вимагаючи від контролера на основі штучної нейронної мережі повертати відповідні з’єднання кінцівки, коли цілі немає в полі зору, і керувати й наближатися до цілі, коли вона є в полі зору. Дана задача надає багато можливостей для ускладнення та варіації, а отже, для вивчення можливостей HyperNEAT у парадигмі навчання з підкріпленням.

Експерименти, які спрямовані на поступове збільшення складності завдання для оцінки ефективності використання HyperNEAT, це:

1. Знаходження і позиціонування кінцівки на задану мінімальну відстань від цільового об’єкта - нерухомої точки на порожній сцені. Для цього потрібно розвертати роборуку на місці, поки ціль не опиниться в полі зору, а потім наближатися прямо до неї.

2. Задача на основі задачі 1, але між ціллю і роботом розташована перешкода, що вимагає вивчення агентом стратегії дослідження для виявлення цільового об’єкту (окрім повороту на місці).

Розміщення перешкоди здійснено так, щоб задачу можна розв’язати за допомогою агента. Експерименти для цих завдань проводилися за допомогою програмного засобу, представленого у розділі 4.3. Зображення з камери генеруються за допомогою бібліотеки pyBullet, яка використовує переваги апаратного прискорення графіки за допомогою OpenGL.

Розглянутий набір задач вимагає вивчення різної поведінки для агента-роборуки, де єдиним входом системи є дані з монокулярної камери у тривимірному середовищі. Аналогічно до двовимірних задач розділу 2, агент

використовує повороти двох ланок роборуки, які виконуються незалежно. Мережі-субстрати мають два виходи, по одному для кожного з'єднання, які масштабуються до діапазону [-180, 180], який вказує на кут повороту відповідного з'єднання. Входом субстрату необроблене зображення камери робота, причому білий колір відповідає значенню 1, а чорний — значенню 0. Цільовий об'єкт зображений на сцені білою сферою. Змодельоване середовище являє собою зону 5×5 метрів. Параметри, які використовуються, перераховані в додатку Д.

4.4.2 Дослідження ефективності використання HyperNEAT для задачі досягнення цільового об'єкта роборукою

Завдання 1: досягнення цільового об'єкта робочою кінцівкою на порожній сцені. Це завдання вимагає від агента знайти цільовий об'єкт та рухати кінцівку до нього, причому ціль нерухома на порожній арені, за обмежений проміжок часу. Агент повинен повертатися, поки не побачить ціль, а потім якомога швидше рухатися до неї, поки не досягне цієї цілі.

Роздільна здатність камери встановлена 25×12 пікселів для мінімізації часу обробки, з кутом огляду 90º по горизонталі (по 45º ліворуч і праворуч). Топологія субстрату являє собою два повнозв'язних шари розміром 25×12 і 2×1. Придатність індивіда визначається за допомогою 8 випробувань. У перших двох випробуваннях початкова орієнтація робота спрямована на 14 градусів ліворуч і 14 градусів праворуч відповідно від прямої лінії до цілі. У чотирьох наступних випробуваннях ці кути були збільшені на 14 градусів, до 42 градусів. Під час останніх двох випробувань робот спочатку дивився на 90 градусів праворуч від прямої лінії до цілі, а потім на 180 градусів. Таким чином, робот повинен був знайти ціль з різних вихідних позицій, і в останніх двох ціль спочатку не видно.

Відстань до цілі в кожному випробуванні регулювалася так, щоб, враховуючи фіксований ліміт часу для випробування, було достатньо часу, щоб розвернутися і направитися приблизно прямо до цілі, а не рухатися, наприклад,

по широкій дузі. Під час останніх двох випробувань агент повинен обертатися за годинниковою стрілкою, поки не знайде ціль.

Функція допасованості базується на відстані, що залишилася до цілі наприкінці кожного випробування для передостаннього кроку часу (використовується передостанній крок як інакше рішення, яке підводить робота надзвичайно близько, але не повністю торкаючись цілі, представляючи це як фактичне торкання цілі, що не дає додаткової допасованості). Зокрема, допасованість f, індивідуума була визначена:

$$f = 1 - \sqrt{\frac{\sum_i (d_i/m)^2}{t}},$$

де d_i — відстань, що залишилася до цілі на передостанньому кроці для спроби i, m — максимально можлива відстань між роботом і ціллю, враховуючи розмір арени, а t — загальна кількість спроб. Ця функція, по суті, є функцією середньоквадратичної помилки, яка в основному усереднює відстань, що залишилася, але забезпечує більшу допасованість для рішень, які наближають кінцівку агента до цілі для більшості випробувань, а не дуже близько в одних випробуваннях і далі в інших.

Коли знайдено рішення, яке могло досягти цілі для кожного тренувального випробування, еволюція зупиняється, і рішення перевіряється 36 разів: у першому випробуванні кінцівка з камерою робота дивилася прямо у напрямку цілі, а в кожному наступному випробуванні робот послідовно обертався 5º (до 180º) проти годинникової стрілки. Крім того, відстань між роботом і ціллю поступово збільшувалася від тієї, що використовувалася під час навчання, до двох разів. Багато початкових поворотів, більшість з яких не узгоджуються з навчальними прикладами, і подальші відстані були перевірені, щоб оцінити ефективність знайденого рішення.

Протестовано різні комбінації та варіації тренувальних випробувань, перш ніж зупинитися на описаному вище. Більшість із них призводили до того, що HyperNEAT застрягав у локальних оптимумах, навіть із великою чисельністю популяції 250 особин і більшою кількістю поколінь -- 1000. Якщо відстані до

цілі не були скориговані для кожного випробування так, щоб було достатньо часу, щоб повернутись і рухатися до цілі, HyperNEAT мав тенденцію застрягати в локальному оптимумі, знаходячи рішення, які б досягли цілі для найпростіших випробувань першими проте унеможливлювали знаходження більш загальних рішень, які могли б досягти цілі для всіх випробувань. Щоб уникнути цієї проблеми недостатньо було змусити функцію припасованості штрафувати менш загальні рішення більше, ніж вона це робила (наприклад, взяти 3-й або 4-й корінь із середнього 3-го або 4-го степеня помилки).

Як показано на рисунку 26 (графік відсотку досягнення цілі), HyperNEAT майже завжди розробляв рішення, яке могло досягти мети для всіх навчальних випробувань. Лише під час останнього випробування (розташування на 180º від цілі) він зазвичай застрягав у локальному оптимумі й обертався не досить швидко, щоб вчасно досягти цілі. Середня відстань, що залишилася за всі випробування, дуже близька до нуля; хоча більшість удосконалених рішень не досягли мети для останнього випробування, вони були дуже близькі до неї, і якщо дати більше часу (ще 5% або 10% від загальної кількості ітерацій), вони досягнуть мети.

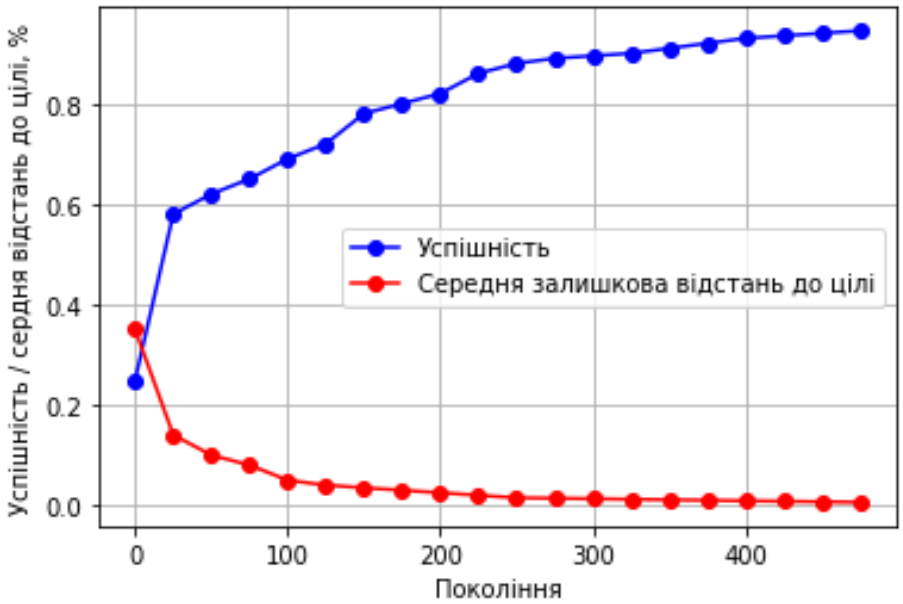


Рис. 26 Продуктивність для завдання 1 досягнення цільового об’єкта агентом-роборукою, усереднена за 30 запусків

Під час тестування представлених моделей приблизно половина змогла ефективно досягти цілі з кожної початкової орієнтації та всіх перевірених відстаней (до вдвічі більшої відстані, ніж була використана під час еволюції). Інша половина рішень працювала до випадків, коли ціль спочатку не була в полі зору, а кінцівка агента була на приблизно в 1,5 рази більшій відстані до цілі (у цих випадках агент також ближче до межі сцени, оскільки агент і ціль поступово переміщувалися вліво і вправо відповідно для кожного наступного тесту). Замість того, щоб продовжувати повертати кінцівку, поки ціль не стала видимою, агент мав тенденцію блукати, поки кінцівка не досягла меж сцени. Таким чином, лише незначної зміни у вхідному зображенні (перебування трохи ближче до границь сцени, поки ціль не було видно) було достатньо, щоб рішення стали неефективними.

Завдання 2: досягнення цільового об'єкта робочою кінцівкою на сцені з перешкодою. Це завдання продовжило попереднє, з додаванням перешкоди між ціллю та агентом, вимагаючи від робота активного пошуку цілі замість того, щоб повертатися на місці, доки ціль не з'явиться в полі зору. Як і раніше, ціль була нерухомою, і робот мав обмежений час, щоб досягти цілі.

Роздільна здатність камери була збільшена до 40×20 пікселів. Топологія субстрату представляє собою три повнозв'язні шари розміром 40×20, 3×3 і 2×1. Перешкода має П-подібну форму, причому її можна розрізнити за формою та інтенсивністю. Придатність індивіда була визначена за допомогою двох випробувань. В обох агент починає пересувати кінцівку з протилежного боку лабіринту до цілі у напрямку камери до неї, але в обох випробуваннях агент і ціль міняються місцями, спочатку з роботом позаду перешкоди, а потім частково всередині нього. Функція допасованості була такою ж, як і для завдання 1.

Подібно до попередньої задачі позиціонування роборуки, виявлено, що HyperNEAT мав тенденцію застрягати в локальних оптимумах. У цьому випадку спочатку були проведені експерименти, у яких використовувався лише перший тест, і було достатньо кроків, щоб кінцівка досягнула мети

оптимальним маршрутом (в експерименті, описаному в методі вище, агент має достатньо часу, не зважаючи на оптимальність маршруту). У цих початкових експериментах HyperNEAT в 75% випадків знаходив рішення, яке спочатку повертало кінцівку перед тим, як обійти лабіринт, але не досягало цілі, оскільки закінчилася кількість кроків через поворот на місці на початку.

Загальне рішення вимагає стежити за стінами перешкоди, утримуючи стіну в частині поля зору (ліворуч або праворуч) і повертаючи кінцівку, якщо стіна не є в полі зору, доки вона не потрапить в поле зору. Оптимальний маршрут полягає у тому, щоб рухатися приблизно прямо до одного з кутів перешкоди, а не повертатися на місці. Неоптимальне рішення виникло тому, що було легше розробити вагові шаблони, які реалізували поведінку слідування за стіною, але також змушували агента повертатися в тому самому напрямку, у якому він міг би тримати стіну частково в полі зору, замість того, щоб повертатися в протилежному напрямку, коли стіна була прямо попереду і займала більшу частину поля зору.

На рисунку 27 показано результати продуктивності для повного експерименту (з використанням двох випробувань, як описано вище). Можна побачити, що HyperNEAT зазвичай знаходить рішення, які могли досягти цілі для одного випробування, але не для обох, однак для одного з 20 прогонів він знайшов рішення, яке могло досягти цілі в обох випробуваннях. Він завжди розвивав рішення, які наближали робота до цілі, тому реалізував стратегію слідування за стіною. У повному експерименті надано достатньо кроків, щоб дозволити розвернути кінцівку перед початком переміщення за перешкоду. Однак HyperNEAT знову застряг у локальному оптимумі. У цьому випадку, коли стратегія слідування за стіною розвинулася, HyperNEAT зазвичай не міг змінити ці рішення, щоб повернути кінцівку до цілі достатньо, коли вона з’явилася в полі зору, щоб досягти її в обох випробуваннях. Замість цього агент зазвичай трохи повертав кінцівку до цілі, але все одно починав переміщуватися повз неї до того, як закінчиться допустима кількість кроків, подібно до стратегії слідування за стіною. В одному з рішень агент повертає кінцівку безпосередньо

до цілі, коли вона потрапляє в поле зору для обох випробувань, тому виявляється, що задача була розв'язана з використанням топології субстратної мережі. Для деяких рішень справедливо наступне: якщо дати трохи більше кроків, то кінцівка досягне цілі, слідуючи короткою траєкторією «згасаючої орбіти».

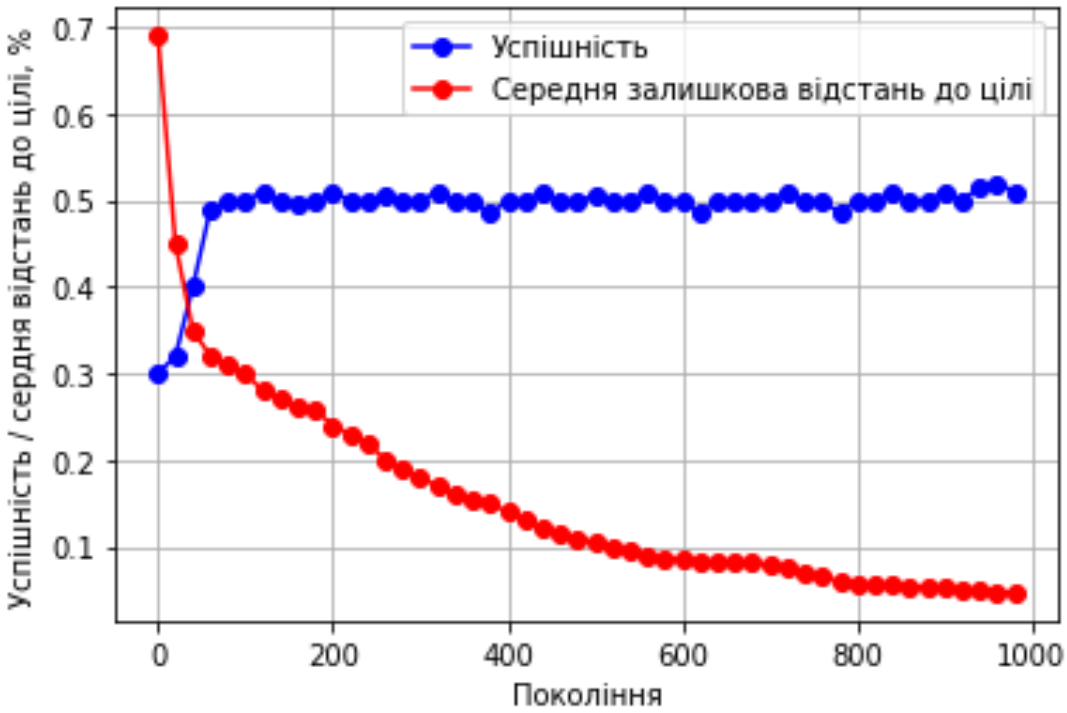


Рис. 27 Продуктивність для завдання 2 досягнення цільового об'єкта агентом-роборукою на сцені з перешкодою, усереднена за 20 запусків

Крім того, просте збільшення частоти мутацій для додавання структурних елементів або зміни значень ваги в CPPN дозволило б HyperNEAT уникнути локального оптимуму.

Метод порівняння для видоутворення при використанні кодування гіперкуба. Виявлено, що в обох завданнях досягнення цільового об'єкта роборукою HyperNEAT мав сильну тенденцію застрягати в локальних оптимумах. Це типова проблема для еволюційних алгоритмів. Компонент NEAT в HyperNEAT (тобто частина методу, яка генерує функції, що кодують ваговий шаблон для субстрату) використовує тип специфікації, що розподіляє припасованість, щоб допомогти уникнути застрягання в локальних оптимумах.

Однак цього, цього виявилося недостатньо для уникнення цієї проблеми для багатьох комбінацій і варіацій тренувальних підходів.

NEAT визначає, чи належить особина до існуючого виду або сумісна з ним за допомогою унікального методу, який по суті порівнює подібність решти генетичної спадщини особини та представника виду. Таким чином чітко підтримується генетична різноманітність. Хоча показано, що видоутворення за допомогою цього методу порівняння покращило продуктивність еволюційного процесу, він явно не ідеальний.

Коли еволюційний алгоритм використовується для розробки вагових функцій для схеми кодування гіперкуба, це відкриває інший можливий спосіб порівняння подібності генотипів, щоб або замінити, або доповнити той, що використовується в NEAT. Два генотипи можна порівняти шляхом прямого порівняння виходу їхніх вагових функцій з різними вхідними значеннями, або, точніше, запитуючи вагові функції з низькою роздільною здатністю в межах діапазону значень, що використовуються в кодуванні гіперкуба. Скалярне значення подібності може бути згенеровано за допомогою, наприклад, функції середньоквадратичної різниці або іншої функції, налаштованої для цього завдання. Перевага такого методу порівняння полягає в тому, що фактичні значення ваги, створені функцією ваги, можуть бути кращим показником поведінки, ніж специфічний генетичний вміст індивідів, і, отже, забезпечувати кращий спосіб підтримувати поведінкову різноманітність, щоб уникнути застрягання в локальних оптимумах. Роздільна здатність, з якою вагові функції запитуються, визначатиме якість порівняння, забезпечуючи масштабований метод якості порівняння в порівнянні з часом, витраченим на виконання порівнянь.

4.5 Висновки до четвертого розділу

Розробка програмних засобів для адаптивного навчання управляючих моделей роборуки на основі нейромереж, що використовують зображення з камери, вимагає комплексного підходу, що має враховувати варіативність умов

середовища, максимально наближених до реального світу. Позиціонування робочої кінцівки має значний простір пошуку, що може бути також ускладнено різним представленням вхідних даних: як набору значень сенсорів, так і вхідним зображенням. Для умов довільного середовища мають бути використані управляючі програмні засоби, що використовують знання, накопичені під час взаємодії з цим середовищем.

Розглянуто основний інструментарій, який має бути представлено у програмному засобі для ефективного дослідження нейроеволюційного підходу до навчання контролерів роборуки. А також представлено розроблений засіб, який забезпечує доступний інтерфейс для роботи з набором середовищ для виконання задач, які представляють основний інтерес у роботі робочої кінцівки робота. Залежно від типу вхідних даних та характеристик агента, програмний засіб надає набір інструментів для налаштування як самого агента та його взаємодії з середовищем, так і параметрів досліджуваного нейроеволюційного алгоритму.

Досліджено ефективність методу HyperNEAT для задач агента-роборуки, що покращують здатність алгоритму знаходити інноваційні та ефективні рішення, сприяючи розвитку більш універсальних і потужних архітектур нейронних мереж.

Встановлено, що розробка оптимальних засобів управління із застосуванням методу HyperNEAT може забезпечити високу ефективність програмного управління, а також мінімальну конфігурацію моделі. Майбутні дослідження з використанням представленого програмного засобу для вирішення задач маніпулювання та позиціонування представляють особливий інтерес для покращення програмних контролерів управління роботизованими кінцівками в динамічних умовах реального світу.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-технічну задачу створення та вдосконалення методів та програмних засобів автоматизованої розробки адаптивних моделей управління роботизованою системою з захватним пристроєм.

В результаті дисертаційного дослідження отримано такі основні наукові результати:

1. Розроблено метод автоматизації створення нейромереж для програмного управління роботизованою кінцівкою з використанням алгоритмів нейроеволюції. Для цього запропоновано метод пошуку новизни в нейроеволюції, що дозволяє прискорити процес розробки програмних засобів управління для нових задач роботизованої кінцівки, що підтверджується за допомогою навчання у тестових середовищах та оцінки ефективності отриманої моделі
2. Розроблено метод підвищення точності системи автоматизованої розробки нейромережевого програмного управління роботизованою кінцівкою з використанням вхідних даних з камери на основі використання методів комп'ютерного зору. Це надає можливість навчання роботизованих агентів на тестових середовищах в умовах, що наближені до реальних, адже враховують похибки камери. Запропонований метод навчання відрізняється від існуючих тим, що дозволяє налаштовувати конфігурацію камери у навчальній системі для адаптації моделі до реальної конфігурації роботизованої системи.
3. Розроблено програмну реалізацію для оцінки впливу похибок параметрів камери на точність програмного управління роботизованою кінцівкою. Здійснено оцінку впливу похибок параметрів калібрування на точність виконання роботизованим агентом вказаного завдання. Встановлено, що підвищення точності

калібрування камер може надати можливість отримати приріст точності тривимірної моделі до 60%

4. Розроблено програмну реалізацію для автоматизованого навчання нейромережевої моделі системи для виконання маніпуляційних задач роботизованою кінцівкою у навчальних середовищах з використанням пошуку новизни. Для задач агента-роборуки, що отримує інформацію про стан середовища з камери, реалізовано навчання з використанням методу на основі гіперкуба.
5. Виконано експериментальні роботи з перевірки ефективності розроблених методів на відомих тестових середовищах OpenAI Gym з використанням реалізованих програмних засобів. Встановлено, що використання пошуку новизни як методу оптимізації нейроеволюційного процесу для вирішення задачі позиціонування дозволяє підвищити ефективність навчального процесу та отримати оптимальну топологію управляючої мережі. для агенту, що використовує зображення з камери, метод на основі гіперкуба покращує ефективність навчання моделі за рахунок використання відображення геометрії фенотипу субстрату штучної нейронної мережі на його шаблон зв’язків на основі гіперкубу, сприяючи розвитку більш універсальних і потужних архітектур нейронних мереж.

Практичне значення одержаних результатів полягає у експериментально підтвердженій ефективності представлених методів на відомих тестових середовищах OpenAI Gym для перевірки якості нейроеволюційних алгоритмів як для двовимірних моделей середовищ, так і для тривимірних, де інформацію про стан середовища роботизований агент отримує з камери, що наближено до роботи системи в умовах реального світу. Отримано патент на засоби калібрування камери для підвищення якості вхідних даних з камери, що використовується для автоматизованого управління роботизованою кінцівкою. Реалізовано метод пошуку новизни під час автоматизованого навчання моделі

системи для двовимірного середовища та метод на основі гіперкуба для тривимірного середовища при виконанні маніпуляційних задач роботизованою кінцівкою. Встановлено, що використання представлених методів у нейроеволюційному процесі для вирішення задачі позиціонування дозволяє підвищити ефективність процесу навчання мережі та отримати оптимальну топологію управляючої мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Пошук новизни у методі нейроеволюції для позиціонування роботизованої кінцівки. Проблеми програмування (3), 49-57 (Фахове видання, “Б”). ISSN 1727-4907
2. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Програмний пакет для адаптивного навчання контролерів роборуки на основі нейромереж. Проблеми програмування (4), 98-107 (Фахове видання, “Б”). ISSN 1727-4907
3. Дорошенко, А. Ю., & Ашур, І. З. (2021). Застосування засобів нейроеволюції в технічних системах автоматизації керування. Проблеми програмування.
4. Вітюк А.Є., Дорошенко А.Ю. (2023). Використання нейроеволюції при пошуку політик в формі нейромереж для управління робочою кінцівкою. Перша Науково-теоретична конференція “Пріоритети і виклики реалізації Стратегії розвитку штучного інтелекту в Україні”. Artificial Intelligence (2). ISSN 2710 – 1673
5. Wurtinger, M. (2011). Neuroevolution for Robot Control. Dissertation. URL: https://www.pst.ifi.lmu.de/Lehre/Abschlussarbeiten/vorlagen/thesis-wuertinger_2011-12-19.pdf
6. Haykin, S. (2009). Neural networks and learning machines, 3/E. Pearson Education India.
7. Norvig, S. R. P. (2010). Artificial Intelligence, A Modern Approach-Prentice Hall.
8. Shademan, A., Decker, R. S., Opfermann, J. D., Leonard, S., Krieger, A., & Kim, P. C. (2016). Supervised autonomous robotic soft tissue surgery. Science translational medicine, 8(337), 64-337
9. Вітюк, А. Є., Корнага, Я. І., & Барабаш, А. О. (2018). Захоплення невідомих об’єктів мобільним роботом із використанням візуальної інформації. Вчені записки Таврійського національного університету

імені В.І. Вернадського. Серія: Технічні науки, (29 (68),№ 1 (1)), 93-98 (Фахове видання, “Б”). ISSN 1606-3721

10. Sancho-Bru, J. L., Mora, M. C., León, B. E., Pérez-González, A., Iserte, J. L., & Morales, A. (2014). Grasp modelling with a biomechanical model of the hand. Computer methods in biomechanics and biomedical engineering, 17(4), 297-310.
11. Han, Y., Wang, B., Koike, H., & Idesawa, M. (2013). Object Recognition with a Limited Database Using Shape Space Theory. In Image Processing: Concepts, Methodologies, Tools, and Applications (pp. 181-200). IGI Global.
12. Laskey, M., Mahler, J., McCarthy, Z., Pokorny, F. T., Patil, S., Van Den Berg, J., ... & Goldberg, K. (2015, August). Multi-armed bandit models for 2d grasp planning with uncertainty. In 2015 IEEE International Conference on Automation Science and Engineering (CASE) (pp. 572-579). IEEE.
13. Laskey, M., Mahler, J., McCarthy, Z., Pokorny, F. T., Patil, S., Van Den Berg, J., ... & Goldberg, K. (2015, August). Multi-armed bandit models for 2d grasp planning with uncertainty. In 2015 IEEE International Conference on Automation Science and Engineering (CASE) (pp. 572-579). IEEE.
14. Popović, M., Kraft, D., Bodenhagen, L., Başeski, E., Pugeault, N., Kragic, D., ... & Krüger, N. (2010). A strategy for grasping unknown objects based on co-planarity and colour information. Robotics and Autonomous Systems, 58(5), 551-565.
15. Bonilla, M., Resasco, D., Gabiccini, M., & Bicchi, A. (2015, September). Grasp planning with soft hands using bounding box object decomposition. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 518-523). IEEE.
16. Bonilla, M., Farnioli, E., Pallottino, L., & Bicchi, A. (2015, May). Sample-based motion planning for robot manipulators with closed kinematic chains. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2522-2527). IEEE.

17. Asfour, T., Borrás, J., Mandery, C., Kaiser, P., Aksoy, E. E., & Grotz, M. (2018). On the dualities between grasping and whole-body loco-manipulation tasks. Robotics Research: Volume 2, 305-322.
18. Pelossof, R., Miller, A., Allen, P., & Jebara, T. (2004, April). An SVM learning approach to robotic grasping. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 (Vol. 4, pp. 3512-3518). IEEE.
19. Huang, B., El-Khoury, S., Li, M., Bryson, J. J., & Billard, A. (2013, May). Learning a real time grasping strategy. In 2013 IEEE International Conference on Robotics and Automation (pp. 593-600). IEEE.
20. Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011, November). DTAM: Dense tracking and mapping in real-time. In 2011 international conference on computer vision (pp. 2320-2327). IEEE.
21. Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In European conference on computer vision (pp. 834-849). Cham: Springer International Publishing.
22. Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, 31(5), 1147-1163.
23. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. The Journal of Machine Learning Research, 22(1), 12348-12355.
24. Вітюк А.Є., Корнага Я.І. (2018). Оцінка впливу похибок калібрування на тривимірну реконструкцію у монокулярній системі одночасної локалізації та картографування. Східноєвропейський науковий журнал (4 (32)), 29–35, ISSN 2468-5380
25. Engel, J., Schöps, T., & Cremers, D. (2014, September). LSD-SLAM: Large-scale direct monocular SLAM. In European conference on computer vision (pp. 834-849). Cham: Springer International Publishing. 10.1007/978-3-319-10605-2_54.

26. Engel, J., Usenko, V., & Cremers, D. (2016). A photometrically calibrated benchmark for monocular visual odometry. arXiv preprint arXiv:1607.02555.
27. Gu, R., Yuan, J., & Zheng, H. (2011, July). Volumetric motion vector calculation in free angle display system. In 2011 Seventh International Conference on Natural Computation (Vol. 2, pp. 987-991). IEEE.
28. Nalpantidis, L., & Gasteratos, A. (2012). Stereo vision depth estimation methods for robotic applications. In Depth Map and 3D Imaging Applications: Algorithms and Technologies (pp. 397-417). IGI global..
29. Zucchelli, M., & Košecká, J. (2008). Motion bias and structure distortion induced by intrinsic calibration errors. Image and Vision Computing, 26(5), 639-646..
30. Yang, S., Bhanu, B., & Mourikis, A. I. (2010, June). Error model for scene reconstruction from motion and stereo. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops (pp. 70-77). IEEE. ISSN 2160-7508.
31. Yang, S., Bhanu, B., & Mourikis, A. I. (2010, June). Error model for scene reconstruction from motion and stereo. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops (pp. 70-77). IEEE.
32. Lehmann, S., Bradley, A. P., Williams, J., Kootsookos, P. J., & Clarkson, L. (2006). Correspondence-free determination of the affine fundamental matrix. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(1), 82-97. ISSN 0162-8828.
33. Belhaoua, A., Kohler, S. & Hirsch, E. J Image Video Proc (2010) 2010: 539836. https://doi.org/10.1155/2010/539836
34. Ghazouani, H., Tagina, M., & Zapata, R. (2011). Robot navigation map building using stereo vision based 3D occupancy grid. Journal of Artificial Intelligence: Theory and Application (JAITA), 1(3), 63-72.
35. Gu, R., Yuan, J., & Zheng, H. (2011, July). Volumetric motion vector calculation in free angle display system. In 2011 Seventh International

Conference on Natural Computation (Vol. 2, pp. 987-991). IEEE. ISSN 2157-9563.

36. Zhang, T., & Boult, T. (2011, January). Realistic stereo error models and finite optimal stereo baselines. In 2011 IEEE Workshop on Applications of Computer Vision (WACV) (pp. 426-433). IEEE. ISSN 1550-5790.
37. Mason, M. T. (2016). Mechanics of robotic manipulation. MIT press.
38. Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). Cnn-slam: Real-time dense monocular slam with learned depth prediction. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6243-6252).
39. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011, May). g 2 o: A general framework for graph optimization. In 2011 IEEE International Conference on Robotics and Automation (pp. 3607-3613). IEEE.
40. Przybylski, M., Asfour, T., & Dillmann, R. (2010, October). Unions of balls for shape approximation in robot grasping. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1592-1599). IEEE.
41. J. E. Goodman and J. O'Rourke, «Handbook of Discrete and Computational Geometry, Second Edition», CRC Press LLC, Boca Raton, FL, 2004.
42. Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012, May). A brief introduction to OpenCV. In 2012 proceedings of the 35th international convention MIPRO (pp. 1725-1730). IEEE.
43. Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395.
44. Xu, B., Jiang, W., Shan, J., Zhang, J., & Li, L. (2015). Investigation on the weighted ransac approaches for building roof plane segmentation from lidar point clouds. Remote Sensing, 8(1), 5.
45. Raguram, R., Chum, O., Pollefeys, M., Matas, J., & Frahm, J. M. (2012). USAC: A universal framework for random sample consensus. IEEE transactions on pattern analysis and machine intelligence, 35(8), 2022-2038.

46. Mahjourian, R., & Miikkulainen, R. (2018). Neuroevolutionary Planning for Robotic Control. Department of Computer Science The University of Texas at Austin Austin.
47. Дорошенко, А. Ю., & Ашур, І. З. (2021). Застосування засобів нейроеволюції в технічних системах автоматизації керування. Проблеми програмування. С. 16-25
48. Huang, P. C., Sentis, L., Lehman, J., Fok, C. L., Mok, A. K., & Miikkulainen, R. (2018). Tradeoffs in neuroevolutionary learning-based real-time robotic task design in the imprecise computation framework. ACM Transactions on Cyber-Physical Systems, 3(2), 1-29.
49. M. W¨urtinger, Neuroevolution for Robot Control, Test Framework and Experimental Evaluation, Institut f¨ur Informatik Lehrstuhl f¨ur Programmierung und Softwaretechnik, 2011. URL: https://www.pst.ifi.lmu.de/Lehre/Abschlussarbeiten/vorlagen/thesis-wuertinger_2011-12-19.pdf
50. OpenAI, O., Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., ... & Zaremba, W. (2021). Asymmetric self-play for automatic goal discovery in robotic manipulation. arXiv preprint arXiv:2101.04882..
51. OpenAI Gym 2D Robot Arm Environment, URL: https://github.com/ekorudiawan/gym-robot-arm
52. Stork, J., Zaefferer, M., Eisler, N., Tichelmann, P., Bartz-Beielstein, T., & Eiben, A. E. (2021, July). Behavior-based neuroevolutionary training in reinforcement learning. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 1753-1761).
53. Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.
54. Huang, P. C., Lehman, J., Mok, A. K., Miikkulainen, R., & Sentis, L. (2014, December). Grasping novel objects with a dexterous robotic hand through neuroevolution. In 2014 IEEE Symposium on Computational Intelligence in

Control and Automation (CICA) (pp. 1-8). IEEE. doi:10.1109/cica.2014.7013242

55. Omelianenko, I. (2019). Hands-On Neuroevolution with Python: Build high-performing artificial neural network architectures using neuroevolution-based algorithms. Packt Publishing Ltd.

56. Вітюк, А. Є., & Дорошенко, А. Ю. (2022). Програмний пакет для оцінки похибки калібрування стереокамери в системі комп'ютерного зору. Проблеми програмування (3-4), 469-477 (Фахове видання, "Б"). ISSN 1727-4907

57. Корнага Я.І., Вітюк А.Є. (2018). Оцінка якості стійкого захвату, спланованого на основі тривимірної реконструкції цільового об'єкту по зображенням з монокамери, Міжнародна науково-технічна конференція "The International Conference on Security, Fault Tolerance, Intelligence".

58. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

59. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

60. Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2015). A survey on evolutionary computation approaches to feature selection. IEEE Transactions on evolutionary computation, 20(4), 606-626.

61. Yang, C., An, Z., Li, C., Diao, B., & Xu, Y. (2019). Multi-objective pruning for cnns using genetic algorithm. In Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part II 28 (pp. 299-305). Springer International Publishing.

62. Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282.

63. Choudhary, T., Mishra, V., Goswami, A., & Sarangapani, J. (2020). A comprehensive survey on model compression and acceleration. Artificial Intelligence Review, 53, 5113-5155.
64. He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622.
65. Zhang, J., & Gouza, F. B. (2018). GADAM: genetic-evolutionary ADAM for deep neural network optimization. arXiv preprint arXiv:1805.07500.
66. Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.
67. He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. Knowledge-Based Systems, 212, 106622..
68. Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y. F., Tu, W. W., ... & Yu, Y. (2018). Taking human out of learning applications: A survey on automated machine learning. arXiv preprint arXiv:1810.13306.
69. Hu, B., Zhao, T., Xie, Y., Wang, Y., Guo, X., Cheng, J., & Chen, Y. (2021, July). MIXP: Efficient Deep Neural Networks Pruning for Further FLOPs Compression via Neuron Bond. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.
70. Jin, Y. (Ed.). (2006). Multi-objective machine learning (Vol. 16). Springer Science & Business Media.
71. Evans, B., Al-Sahaf, H., Xue, B., & Zhang, M. (2018, July). Evolutionary deep learning: A genetic programming approach to image classification. In 2018 IEEE congress on evolutionary computation (CEC) (pp. 1-6). IEEE.
72. Telikani, A., Tahmassebi, A., Banzhaf, W., & Gandomi, A. H. (2021). Evolutionary machine learning: A survey. ACM Computing Surveys (CSUR), 54(8), 1-35.
73. Zhang, M. (2018). Evolutionary deep learning for image analysis. In Talk at World Congress on Computational Intelligence (WCCI).
74. Zhang, M., & Cagnoni, S. (2020, July). Evolutionary computation and evolutionary deep learning for image analysis, signal processing and pattern

recognition. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion (pp. 1343-1372).

75. Tran, B., Xue, B., & Zhang, M. (2017). A new representation in PSO for discretization-based feature selection. IEEE Transactions on Cybernetics, 48(6), 1733-1746.
76. Luo, R., Tian, F., Qin, T., Chen, E., & Liu, T. Y. (2018). Neural architecture optimization. Advances in neural information processing systems, 31.
77. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... & Hodjat, B. (2024). Evolving deep neural networks. In Artificial intelligence in the age of neural networks and brain computing (pp. 269-287). Academic Press.
78. Dowdell, T., & Zhang, H. (2020). Language modelling for source code with transformer-xl. arXiv preprint arXiv:2007.15813.
79. Ma, L., Huang, M., Yang, S., Wang, R., & Wang, X. (2021). An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. IEEE Transactions on Cybernetics, 52(7), 6684-6696.
80. Vargas-Hakim, G. A., Mezura-Montes, E., & Acosta-Mesa, H. G. (2021). A review on convolutional neural network encodings for neuroevolution. IEEE Transactions on Evolutionary Computation, 26(1), 12-27.
81. Miahi, E., Mirroshandel, S. A., & Nasr, A. (2022). Genetic Neural Architecture Search for automatic assessment of human sperm images. Expert Systems with Applications, 188, 115937.
82. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., ... & Kurakin, A. (2017, July). Large-scale evolution of image classifiers. In International conference on machine learning (pp. 2902-2911). PMLR.
83. Vafaie, H., & De Jong, K. (1998). Feature space transformation using genetic algorithms. IEEE Intelligent Systems and their Applications, 13(2), 57-65.

84. Ren, P., Xiao, Y., Chang, X., Huang, P. Y., Li, Z., Chen, X., & Wang, X. (2021). A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys (CSUR), 54(4), 1-34.
85. Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 50(9), 3840-3854.
86. Yao, X. (1993). A review of evolutionary artificial neural networks. International journal of intelligent systems, 8(4), 539-567.
87. Li, N., Ma, L., Yu, G., Xue, B., Zhang, M., & Jin, Y. (2023). Survey on evolutionary deep learning: Principles, algorithms, applications, and open issues. ACM Computing Surveys, 56(2), 1-34. https://doi.org/10.1145/3603704
88. Evolutionary Deep Learning: Genetic algorithms and neural networks. Author, Michael Lanham. Publisher, Manning, 2023. ISBN, 1617299529
89. Lehman, J., & Stanley, K. O. (2013, November). Exploring Biological Intelligence through Artificial Intelligence and Radical Reimplementation. In 2013 AAAI Fall Symposium Series. https://stars.library.ucf.edu/scopus2010/7636
90. Kroeger, O., Huegle, J., & Niebuhr, C. A. (2019, September). An automatic calibration approach for a multi-camera-robot system. In 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1515-1518). IEEE. doi:10.1109/etfa.2019.8869522
91. Risi, S., & Stanley, K. O. (2011, July). Enhancing es-hyperneat to evolve more complex regular neural networks. In Proceedings of the 13th annual conference on Genetic and evolutionary computation (pp. 1539-1546).
92. Coleman, O. J. (2010). Evolving neural networks for visual processing. Undergraduate Honours Thesis (Bachelor of Computer Science, University of New South Wales School of Computer Science and Engineering).
93. van den Berg, T. G., & Whiteson, S. (2013, July). Critical factors in the performance of HyperNEAT. In Proceedings of the 15th annual conference on

Genetic and evolutionary computation (pp. 759-766). 10.1145/2463372.2463460.

94. Gauci, J., & Stanley, K. (2007, July). Generating large-scale neural networks through discovering geometric regularities. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (pp. 997-1004).
95. Stanley, K. O. (2006). Exploiting Regularity Without Development. In AAAI Fall Symposium: Developmental Systems (p. 49).
96. D'Ambrosio, D. B., & Stanley, K. O. (2007, July). A novel generative encoding for exploiting neural network sensor and output geometry. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (pp. 974-981).
97. Jaskowski, W., Krawiec, K., & Wieloch, B. (2008, October). Neurohunter-an entry for the balanced diet contest. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008) Contest Program".
98. Alona Vitiuk, Anatoliy Doroshenko. (2022) Software Package for Evaluation the Stereo Camera Calibration for 3D Reconstruction in Robotics Grasping System. CEUR Workshop Proceedings Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG, ISSN 1613-0073
99. US10841570B2, Petro Kytsun Iegor Vdovychenko, Alona Vitiuk, Nataliya Sakhnenko, Oleksii Panfilov, «Simultaneous camera and sensors calibration apparatus and method», 2018

ДОДАТКИ

Додаток А. Акти про впровадження результатів дисертаційної роботи

АКТ

про впровадження результатів дисертаційної роботи
Вітюк Альони Євгеніївни

Комісія в складі: голови комісії директора ТОВ Інтехфорвард Є.В. Крилова, членів комісії: **менеджера** С.І. Крилової та **інженера** І.І. Казмирчука склала цей акт про те, що результати, отримані в дисертаційному дослідженні аспірантки Національного технічного університету «Київський політехнічний інститут імені Ігоря Сікорського» Вітюк Альони Євгеніївни, реалізовані при впровадженні розробки автоматизованого управління роботизованими системами. Тема дисертації «Методи і програмні засоби для автоматизації управління роботизованою кінцівкою».

Прийняті до впровадження такі результати дослідження: метод пошуку новизни в нейроеволюції для автоматизації програмного управління роботизованою кінцівкою; метод підвищення якості вхідних даних з камери, які використовуються для автоматизованого управління роботизованою кінцівкою, що надає можливість навчання роботизованих агентів на тестових середовищах в умовах, що наближені до реальних; алгоритм для оцінки впливу похибок параметрів камери на точність роботизованої системи.

Імітаційна математична модель роботизованої системи відтворила поведінку захватного пристрою у різних умовах та сценаріях управління, підтверджуючи ефективність та стійкість запропонованих алгоритмів.

Даний акт не є підставою для фінансових взаєморозрахунків.

Голова комісії:
Директор ТОВ Інтехфорвард _____ Є.В. Крилов

Члени комісії:
Менеджер _____ С.І. Крилова

Інженер _____ І.І. Казмирчук

«ЗАТВЕРДЖУЮ»
Декан факультету інформатики
та обчислювальної техніки
КПІ ім. Ігоря Сікорського

_____ Я.І. Корнага
«___» _____ 2023 р.

АКТ

про впровадження в навчальний процес кафедри Інформаційних систем та технологій факультету Інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» результатів дисертаційної роботи Вітюк Альони Євгеніївни «Методи і програмні засоби для автоматизації управління роботизованою кінцівкою», поданої на здобуття наукового ступеня доктора філософії.

Ми, що нижче підписалися: завідувач кафедри Інформаційних систем та технологій, д.т.н. проф. Ролік О.І., доцент кафедри інформаційних систем та технологій, к.т.н., доц. Пасько В.П., вчений секретар кафедри, к.ф.-м.н., доц. Гавриленко О.В. розглянувши матеріали навчально-методичного забезпечення курсів: «Спеціальні розділи математики», «Бази даних», «Системне програмування», «Аналіз даних в інформаційно-управляючих системах» які підготовлені та викладаються асистенткою Вітюк А.Є., встановили, що в навчальному процесі на лабораторних заняттях, а також в процесі підготовки магістерських і бакалаврських атестаційних робіт використовуються такі наукові результати, отримані дисертантом особисто:

- алгоритм застосування нейроеволюції, що дозволяє автоматизувати навчання управляючої нейромережі для використання при розробці програмних рішень аналізу даних;
- механізми застосування алгоритмів, які дозволяють виконувати автоматизоване навчання нейкроконтролерів прикладних систем;

- інформаційна технологія для автоматизації процесів розробки програмного коду.

Впровадження результатів дисертаційної роботи Вітюк А.Є. в навчальний процес дозволило підвищити якість підготовки студентів освітнього рівня «Бакалавр» спеціальностей 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології» на кафедрі Інформаційних систем та технологій КПІ ім. Ігоря Сікорського.

Завідувач кафедри
інформаційних систем та технологій,
д.т.н., проф. О.І. Ролік

Доцент кафедри
інформаційних систем та технологій,
к.т.н., доц. В.П. Пасько

Вчений секретар кафедри
інформаційних систем та технологій,
к.т н., доц. О.В. Гавриленко

Додаток Б. Програмний код

```
import os, sys
import argparse
from argparse import ArgumentDefaultsHelpFormatter
import pickle
import warnings
import multiprocessing as mp

import neat
 from neat_gym.novelty import Novelty

sys.path.insert(0, '.')

from tools.gym_neat_config import GymNeatConfig
from tools.gym_hyper_config import GymHyperConfig
from tools.gym_population import GymPopulation
from tools.reporter import SaveReporter, _StdOutReporter

parser = argparse.ArgumentParser(
        formatter_class=ArgumentDefaultsHelpFormatter)
group = parser.add_mutually_exclusive_group()
parser.add_argument('configfile', metavar='CONFIGFILE',
                    help='input config file')
group.add_argument('–is_hyper', action='store_true', help='Use HyperNEAT')
group.add_argument('–is_eshyper', action='store_true',
                   help='Use ES-HyperNEAT')
parser.add_argument('–is_novelty', action='store_true',
                    help='Use Novelty Search')
parser.add_argument('--maxtime', default=None, type=int,
                    help='Maximum time in seconds')
args = parser.parse_args(["config/kuka", "--hyper"])

if args.is_hyper:
    config = GymHyperConfig(args)
if args.is_eshyper:
    config = GymEsHyperConfig(args)
else:
    config = GymNeatConfig(args)
```

```
stats = neat.StatisticsReporter()

pop = (NoveltyPopulation(config, stats)
       if config.is_novelty()
       else GymPopulation(config, stats))

pop.add_reporter(_StdOutReporter(show_species_detail=False))
pop.add_reporter(stats)

pop.add_reporter(SaveReporter(config.env_name,
                              config.checkpoint,
                              args.novelty))

print("ParallelEvaluator cpu count: ", mp.cpu_count())
pe = neat.ParallelEvaluator(mp.cpu_count(), config.eval_genome, timeout=20)

winner = pop.run(pe.evaluate, config.ngen, args.maxtime, config)

config.save_genome(winner)
```

Пакет tools, модуль gym_neat_config

```
from configparser import ConfigParser

import os
import random
import pickle
import neat
from neat.nn import FeedForwardNetwork
from neat.math_util import mean, stdev
from neat.config import ConfigParameter

from neat_gym import _gym_make, _is_discrete, eval_net
from pureples.shared.visualize import draw_net

class GymNeatConfig(object):
    parameters = [ConfigParameter('pop_size', int),
                  ConfigParameter('reset_on_extinction', bool),
                  ConfigParameter('fitness_criterion', str),
                  ConfigParameter('fitness_threshold', float),
                  ConfigParameter('no_fitness_termination', bool, False)]
```

```
def __init__(self, args, layout=None):

    if not os.path.isfile(args.configfile):
        print('Config file does not exist: %s' %
              os.path.abspath(args.configfile))
        exit(1)

    self.genome_type = neat.DefaultGenome
    self.reproduction_type = neat.DefaultReproduction
    self.species_set_type = neat.DefaultSpeciesSet
    self.stagnation_type = neat.DefaultStagnation

    parameters = ConfigParser()
    with open(args.configfile) as f:
        if hasattr(parameters, 'read_file'):
            parameters.read_file(f)
        else:
            parameters.readfp(f)

        self.node_names = {}

        try:
            names = parameters['Names']
            for idx, name in enumerate(eval(names['input'])):
                self.node_names[-idx-1] = name
            for idx, name in enumerate(eval(names['output'])):
                self.node_names[idx] = name
        except Exception:
            pass

    param_list_names = []

    for p in self.__params:
        if p.default is None:
            setattr(self, p.name, p.parse('NEAT', parameters))
        else:
            try:
                setattr(self, p.name, p.parse('NEAT', parameters))
            except Exception:
                setattr(self, p.name, p.default)
```

```
            warnings.warn('Using default %s for %s' %
                          (p.default, p.name), DeprecationWarning)
    param_list_names.append(p.name)

self.check_params(args.configfile, parameters, 'NEAT')
self.check_params(args.configfile, parameters, 'Gym')

gympar = parameters['Gym']
env_name = gympar['environment']
self.reps = int(gympar['episode_reps'])

env = gym_make(env_name)

if layout is None:
    num_inputs = env.observation_space.shape[0]
    #num_inputs = env.observation_space['observation'].shape[0]
    if _is_discrete(env):
        num_outputs = env.action_space.n
    else:
        num_outputs = env.action_space.shape[0]
else:
    num_inputs, num_outputs = layout
print('num_inputs is ', num_inputs)
print('num_outputs is ', num_outputs)
genome_dict = dict(parameters.items(self.genome_type.__name__))

genome_dict['num_inputs'] = num_inputs
genome_dict['num_outputs'] = num_outputs

self.genome_config = self.genome_type.parse_config(genome_dict)

stagnation_dict = dict(parameters.items(self.stagnation_type.__name__))
self.stagnation_config = \
    self.stagnation_type.parse_config(stagnation_dict)

self.species_set_dict = \
    dict(parameters.items(self.species_set_type.__name__))
self.species_set_config = \
    self.species_set_type.parse_config(self.species_set_dict)

self.reproduction_dict = \
```

```
            dict(parameters.items(self.reproduction_type.__name__))
        self.reproduction_config = \
            self.reproduction_type.parse_config(self.reproduction_dict)

        self.env_name = env_name

        neatparams = parameters['NEAT']
        self.checkpoint = self.get_with_default(neatparams, 'checkpoint',
                                                lambda s: bool(s), False)
        self.ngen = self.get_with_default(neatparams, 'generations',
                                          lambda s: int(s), None)
        self.seed = self.get_with_default(neatparams, 'seed',
                                          lambda s: int(s), None)

        random.seed(self.seed)

        self.max_episode_steps = env.spec.max_episode_steps
        if self.max_episode_steps is None:
            self.max_episode_steps = 400

        self.env = env

        self.current_evaluations = 0
        self.total_evaluations = 0

        self.novelty = GymNeatConfig.parse_novelty(args.configfile) \
            if args.novelty else None

        self.params = parameters

        self.gen = 0

        self.activations = 1

    def evaluate_net_mean(self, net, genome):

        return (self.evaluate_net_mean_novelty(net, genome)
                if self.is_novelty()
                else self.eval_net_mean_reward(net, genome))

    def evaluate_net_mean_reward(self, net, genome):
```

```
        reward_sum = 0
        total_steps = 0

        for _ in range(self.reps):
            reward, steps = eval_net(net,
                                     self.env,
                                     activations=self.activations,
                                     seed=self.seed,
                                     max_episode_steps=self.max_episode_steps,
                                     report=False)
                                     #csvfilename='eval_log.csv')

            reward_sum += reward
            total_steps += steps

        return reward_sum/self.reps, total_steps

    def evaluate_net_mean_novelty(self, net, genome):
        reward_sum = 0
        total_steps = 0

        behaviors = [None] * self.reps

        for j in range(self.reps):

            reward, behavior, steps = self.evaluate_net_novelty(net, genome)
            reward_sum += reward
            behaviors[j] = behavior
            total_steps += steps

        return reward_sum/self.reps, behaviors, total_steps

    def evaluate_net_novelty(self, net, genome):
        env = self.env
        env.seed(self.seed)
        state = env.reset()
        steps = 0

        is_discrete = _is_discrete(env)

        total_reward = 0
```

```
        while self.max_episode_steps is None or steps < self.max_episode_steps:

            for k in range(self.activations):
                action = net.activate(state)

            action = (np.argmax(action)
                      if is_discrete
                      else action * env.action_space.high)

            state, reward, done, info = env.step(action)

            behavior = info['curr_posn']

            total_reward += reward

            if done:
                break

            steps += 1
        if steps == self.max_episode_steps:
            total_reward = total_reward - 10
        env.close()

        return total_reward, behavior, steps

    def save_genome(self, genome, count=0):

        name = self.make_name(genome, count)
        net = FeedForwardNetwork.create(genome, self)
        pickle.dump((net, self.env_name), open('models/%s.dat' % name, 'wb'))
        GymNeatConfig.draw_net(net,
                               'visuals/%s-network' % name,
                               self.node_names)

    def is_novelty(self):

        return self.novelty is not None

    def make_name(self, genome, count=0, suffix=''):
```

```
        return '%s_s%s_c%d_f%+010.3f' % \
               (self.env_name, suffix, count, genome.actual_fitness)

    def get_with_default(self, params, name, fun, default):
        return fun(params[name]) if name in params else default

    def check_params(self, filename, params, section_name):
        if not params.has_section(section_name):
            self.error('%s section missing from configuration file %s' %
                       (section_name, filename))

    def error(self, msg):
        print('ERROR: ' + msg)
        exit(1)

    def draw_network(net, filename, node_names):
        draw_network(net, filename=filename, node_names=node_names)

        os.remove(filename)

    @staticmethod
    def evaluate_genome(genome, config):
        print("eval_genome started")
        net = FeedForwardNetwork.create(genome, config)
        return config.eval_net_mean(net, genome)

    @staticmethod
    def parse_novelty(cfgfilename):

        novelty = None

        parameters = ConfigParser()

        with open(cfgfilename) as f:
            if hasattr(parameters, 'read_file'):
                parameters.read_file(f)
            else:
                parameters.readfp(f)

            try:
                names = parameters['Novelty']
```

```
                novelty = Novelty(eval(names['k']),
                                  eval(names['threshold']),
                                  eval(names['limit']),
                                  eval(names['ndims']))
            except Exception:
                print('File %s has no [Novelty] section' % cfgfilename)
                exit(1)

        return novelty
```

Пакет tools, модуль gym_population

```
from time import time
import numpy as np
import matplotlib.pyplot as plt
import neat
from neat.population import Population, CompleteExtinctionException

class GymPopulation(Population):
    def __init__(self, config, stats):

        Population.__init__(self, config)

        self.config = config

        self.stats = stats

    def run(self, fitness_function, ngen, maxtime, config):

        print("maxtime is ", maxtime)
        gen = 0
        start = time()

        count = 0

        while ((ngen is None or gen < ngen)
               and (maxtime is None or time()-start < maxtime)):

            self.config.gen = gen

            gen += 1
```

```
self.config.current_evaluations = 0

self.reporters.start_generation(self.generation)
try:
    for i, p in self.population.items():
        print("Pop is ", type(i), type(p))
    fitness_function(list(self.population.items()), self.config)
except:
    print("fitness_function call failed!")
    continue
print("fitness_function succeeded")

best = None
for g in self.population.values():

    if g.fitness is None:
        raise RuntimeError('Fitness not assigned to genome %d' %
                           g.key)
    g.fitness, g.actual_fitness, evaluations = (
            self.parse_fitness(g.fitness))
    self.config.current_evaluations += evaluations
    self.config.total_evaluations += evaluations
    if best is None:
        best = g

    else:
        if g.actual_fitness > best.actual_fitness:
            best = g

self.reporters.post_evaluate(self.config,
                             self.population,
                             self.species,
                             best)

if (self.best_genome is None or
        best.actual_fitness > self.best_genome.actual_fitness):
    self.best_genome = best

if not self.config.no_fitness_termination:
    fv = self.fitness_criterion(g.actual_fitness
```

```
                                    for g in self.population.values())
            if fv >= self.config.fitness_threshold:
                self.reporters.found_solution(self.config,
                                              self.generation,
                                              best)
                break

        self.reproduce()

        if not self.species.species:
            self.reporters.complete_extinction()

            if self.config.reset_on_extinction:
                self.create_new_pop()
            else:
                raise CompleteExtinctionException()

        self.species.speciate(self.config,
                              self.population,
                              self.generation)

        self.reporters.end_generation(self.config,
                                      self.population,
                                      self.species)

        self.generation += 1

        if(count%100 == 0):
            config.save_genome(self.best_genome, count)
            self.plot_species()
        count += 1

    if self.config.no_fitness_termination:
        self.reporters.found_solution(self.config,
                                      self.generation,
                                      self.best_genome)

    self.plot_species()

    return self.best_genome
```

```
    def reproduce(self):
        self.population = \
                 self.reproduction.reproduce(self.config, self.species,
                                             self.config.pop_size,
                                             self.generation)

    def create_new_pop(self):
        self.population = \
                self.reproduction.create_new(self.config.genome_type,
                                             self.config.genome_config,
                                             self.config.pop_size)

    def parse_fitness(self, fitness):
        return fitness[0], fitness[0], fitness[1]

    def plot_species(self):
        species_sizes = self.stats.get_species_sizes()
        num_generations = len(species_sizes)
        curves = np.array(species_sizes).T

        fig, ax = plt.subplots()
        ax.stackplot(range(num_generations), *curves)

        filename = self.config.make_name(self.best_genome)

        plt.title(filename)
        plt.ylabel("Size per Species")
        plt.xlabel("Generations")

        plt.savefig('visuals/%s-species.pdf' % filename)
        print("savefile ", 'visuals/%s-species.pdf' % filename)

        plt.close()
```

Пакет tools, модуль novelty_population

```
from gym_population import *

class NoveltyPopulation(GymPopulation):

    def __init__(self, config, stats):
```

```
        GymPopulation.__init__(self, config, stats)

    def parse_fitness(self, fitness):
        actual_fitness, behaviors, evaluations = fitness

        fitness = np.sum([0 if behavior is None
                          else self.config.novelty.add(behavior)
                          for behavior in behaviors])

        return fitness, actual_fitness, evaluations
```

Додаток В. Патент US10841570B2, Petro Kytsun Iegor Vdovychenko, Alona Vitiuk, Nataliya Sakhnenko, Oleksii Panfilov, «Simultaneous camera and sensors calibration apparatus and method»

US010841570B2

(12) **United States Patent**
Kytsun et al.

(10) **Patent No.: US 10,841,570 B2**
(45) **Date of Patent: Nov. 17, 2020**

(54) **CALIBRATION DEVICE AND METHOD OF OPERATING THE SAME**

(71) Applicant: **Samsung Electronics Co., Ltd.**, Suwon-si, Gyeonggi-do (KR)

(72) Inventors: **Petro Kytsun**, Kyiv (UA); **Iegor Vdovychenko**, Kharkov (UA); **Alona Vitiuk**, Kyiv (UA); **Nataliya Sakhnenko**, Kharkov (UA); **Oleksii Panfilov**, Kyiv (UA)

(73) Assignee: **Samsung Electronics Co., Ltd.**, Suwon-si (KR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/407,707**

(22) Filed: **May 9, 2019**

(65) **Prior Publication Data**

US 2020/0014912 A1 Jan. 9, 2020

(30) **Foreign Application Priority Data**

Jul. 6, 2018 (KR) 10-2018-0078930

(51) **Int. Cl.**
G01P 13/00 (2006.01)
H04N 17/00 (2006.01)
G06T 7/80 (2017.01)

(52) **U.S. Cl.**
CPC ***H04N 17/002*** (2013.01); ***G01P 13/00*** (2013.01); ***G06T 7/80*** (2017.01)

(58) **Field of Classification Search**
CPC .. H04N 13/246; H04N 13/296; H04N 5/2253; H04N 5/23258; H04N 5/23267;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,209,383 B1 4/2001 Mueller et al.
8,619,144 B1 12/2013 Chang et al.
(Continued)

OTHER PUBLICATIONS

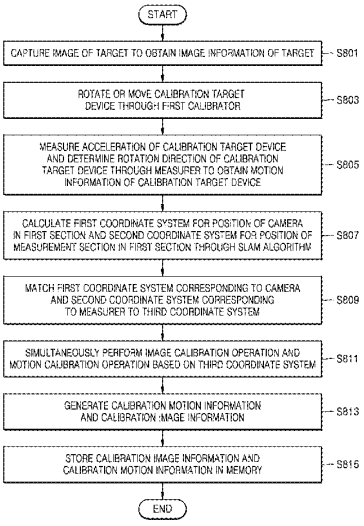
Gerard Medioni, Sing Bing Kang, Emerging topics in computer vision. Camera calibration, Chapter 2; Prentice Hall PTR Upper Saddle River, NJ, USA ©2004, ISBN:0131013661.
(Continued)

Primary Examiner — Paulos M Natnael
(74) *Attorney, Agent, or Firm* — Jefferson IP Law, LLP

(57) **ABSTRACT**

A calibration method is provided. The calibration method includes capturing an image of a target, acquiring image information of the target, acquiring motion information of a calibration target device by measuring an acceleration of the calibration target device and determining a rotation direction of the calibration target device, matching a first coordinate system corresponding to a camera configured to capture the image of the target and a second coordinate system corresponding to a measurer configured to acquire the motion information of the calibration target device to a third coordinate system, simultaneously performing, based on the third coordinate system, an image calibration operation with respect to the image information and a motion calibration operation with respect to the motion information, generating calibration motion information via the image calibration operation with respect to the image information, and generating calibration image information via the motion calibration operation with respect to the motion information.

19 Claims, 11 Drawing Sheets



(58) **Field of Classification Search**
CPC H04N 5/2327; B25J 9/1697; G03B 2207/005
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

9,683,837	B2	6/2017	Siercks	
2003/0038933	A1*	2/2003	Shirley	G01B 11/25 356/243.1
2007/0075997	A1*	4/2007	Rohaly	G06T 7/80 345/419
2007/0229665	A1*	10/2007	Tobiason	H04N 5/3572 348/187
2011/0026014	A1*	2/2011	Mack	G03B 13/22 356/124
2014/0218550	A1*	8/2014	Chuang	H04N 5/23248 348/208.6
2015/0341531	A1*	11/2015	Senda	H04N 5/23258 348/308
2017/0019656	A1	1/2017	Liu et al.	
2017/0032537	A1*	2/2017	Li	H04N 17/002
2017/0039715	A1*	2/2017	Yoshioka	H04N 17/002
2017/0287166	A1*	10/2017	Claveau	G06T 7/80

OTHER PUBLICATIONS

O.S. Kireyev, Automotive calibration of a stereo pair in laboratory environment; Math. Machines and systems, 2004, No. 1. p. 86-100.
Sebastian O.H. Madgwick, "Automated Calibration of an accelerometers, Magnetometers and Gyroscopes—A feasibility study," Sep. 20, 2010.
Bouguet J.Y., "Camera Calibration Toolbox for Matlab", California Institute of Technology, Pasadena, CA, US online at http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.; Oct. 14, 2015.
R. Hartley & A. Zisserman, "Multiple view geometry in computer vision", chapter 4.1 p. 89, Cambridge University Press 2000, 2003.
Juho Kannala and Sami S. Brandt, "A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses" IEEE Transactions on Pattern Analysis and Machine Intelligence, V. 28, Aug. 2006.
Rainer Kuemmerle et al., "g2o, A General Framework for Graph Optimization", IEEE International Conference on Robotics and Automation (ICRA), 2011.
Google, 3D-cameras on smartphones for VR applications (Google Project Tango, http://podrobnosti.ua/2083006-google-sozdal-smartfon-s-3d-kameroj-video.html), 2016.
Depth sensors (http://scientificrussia.ru/articles/polyarizatsiya-sveta-pozvolit-uvelichit-razreshenie-3d-kamer-v-tysyachu-raz), 2015.
3D laser displays (http://www.dailytechinfo.org/infotech/7462-kompaniya-trilite-technologies-gotovit-nachalo-proizvodstva-ogromnyh-trehmemyh-displeev-v-stile-nazad-v-buduschee.html), 2015.

* cited by examiner

FIG. 1A

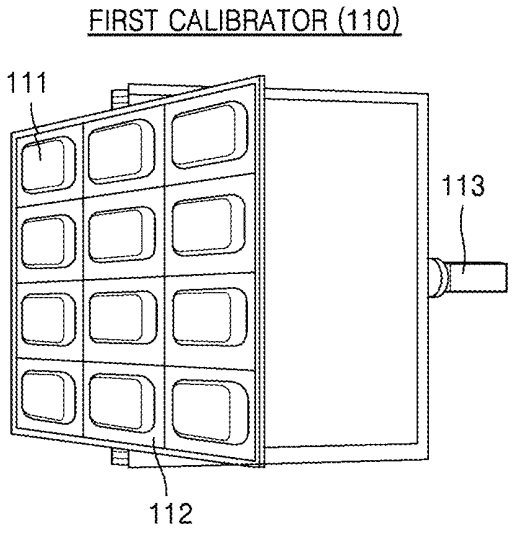


FIG. 1B

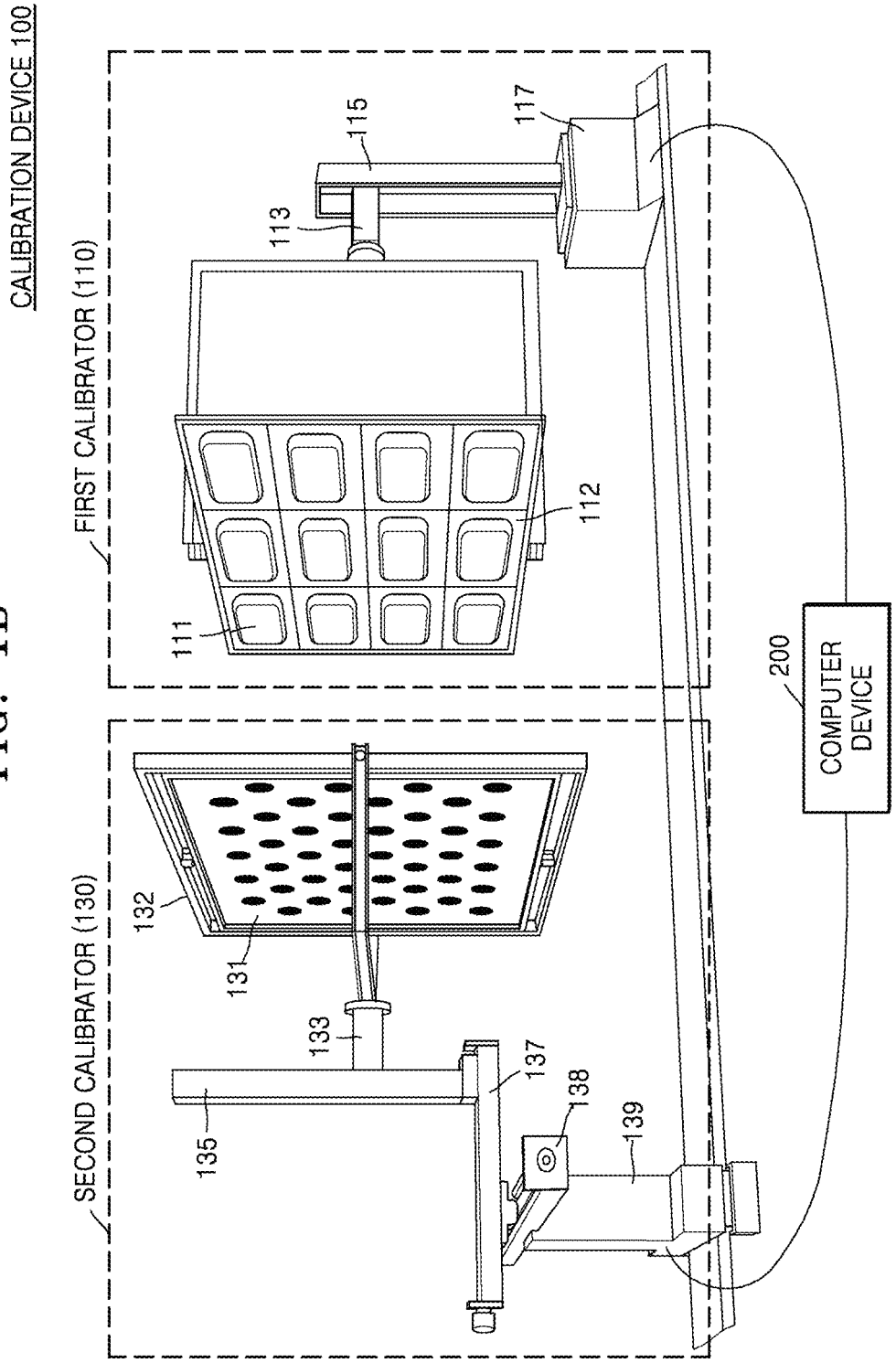

CALIBRATION DEVICE 100
FIRST CALIBRATOR (110)
SECOND CALIBRATOR (130)
111
112
113
115
117
131
132
133
135
137
138
139
200
COMPUTER DEVICE

FIG. 2

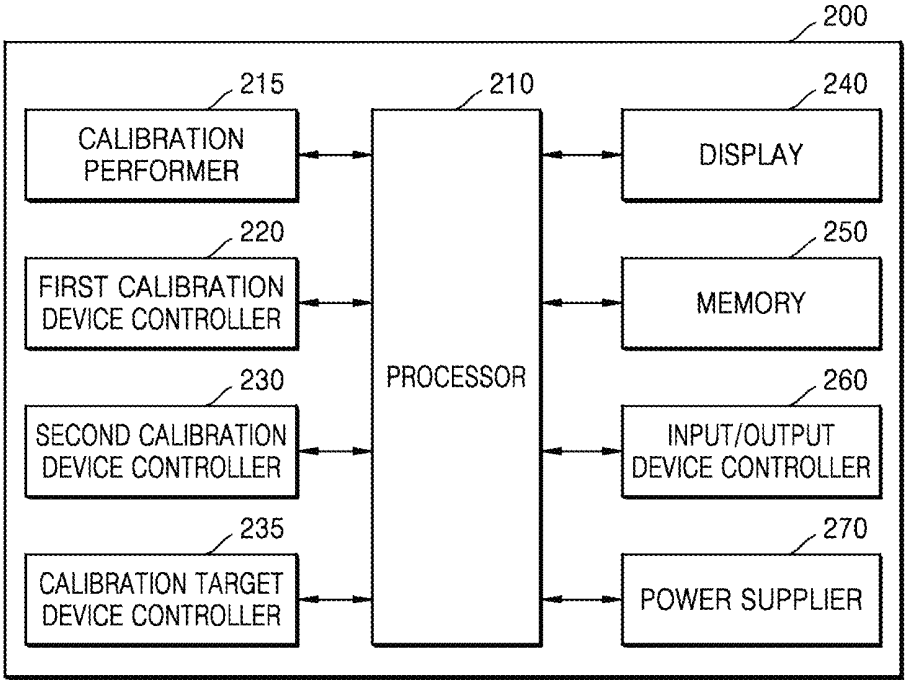
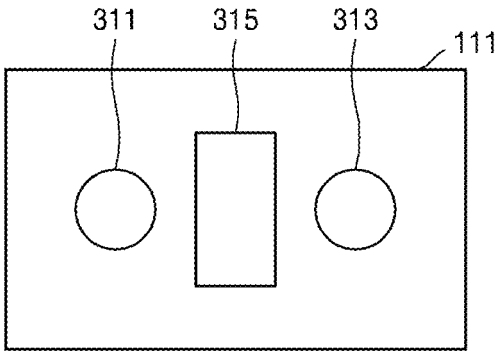
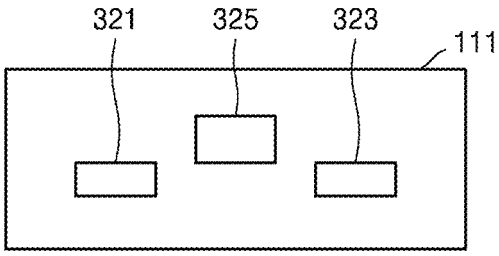

200
215
CALIBRATION PERFORMER
210
PROCESSOR
240
DISPLAY
220
FIRST CALIBRATION DEVICE CONTROLLER
250
MEMORY
230
SECOND CALIBRATION DEVICE CONTROLLER
260
INPUT/OUTPUT DEVICE CONTROLLER
235
CALIBRATION TARGET DEVICE CONTROLLER
270
POWER SUPPLIER

FIG. 3A



FRONT VIEW

FIG. 3B



TOP VIEW

FIG. 4

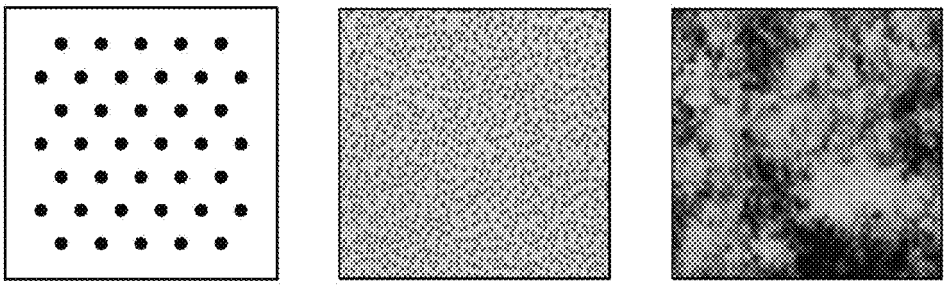


FIG. 5A

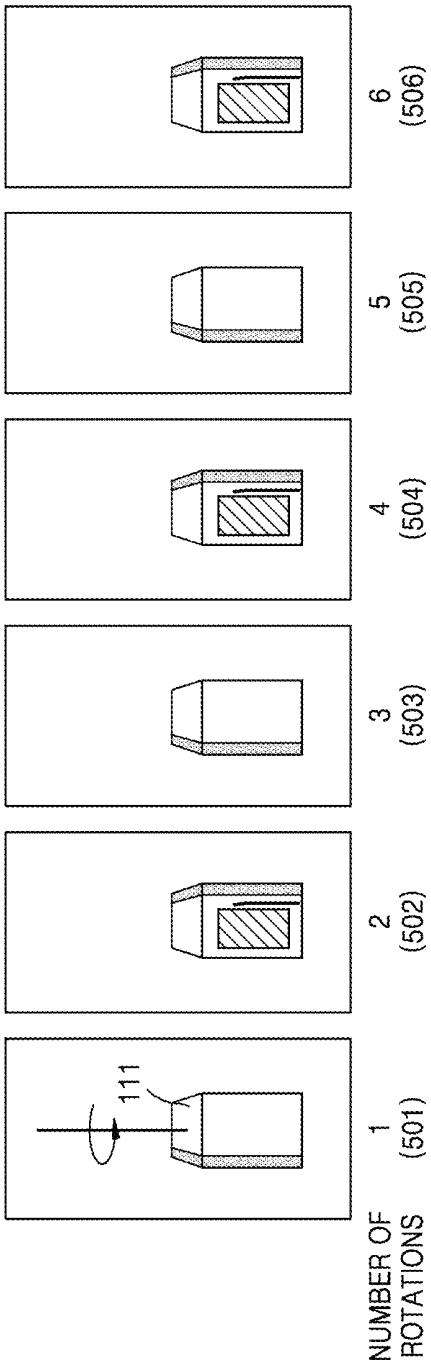


FIG. 5B

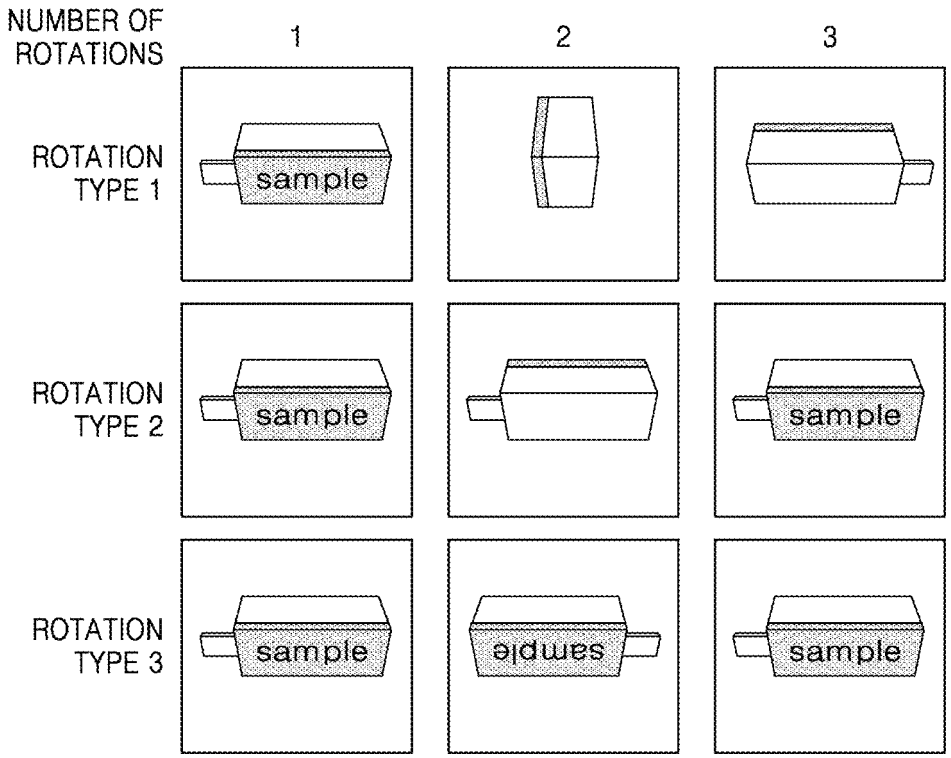

NUMBER OF ROTATIONS
1
2
3
ROTATION TYPE 1
ROTATION TYPE 2
ROTATION TYPE 3
sample
sample
sample
sample
sample
sample

FIG. 6

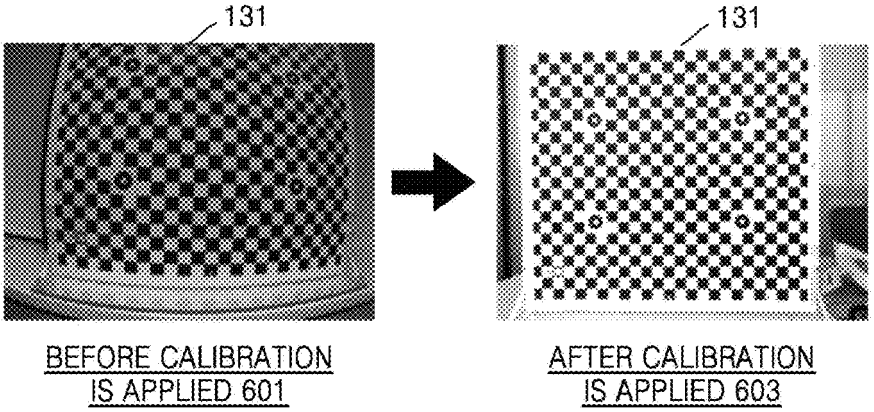

131
131
BEFORE CALIBRATION
IS APPLIED 601
AFTER CALIBRATION
IS APPLIED 603

FIG. 7

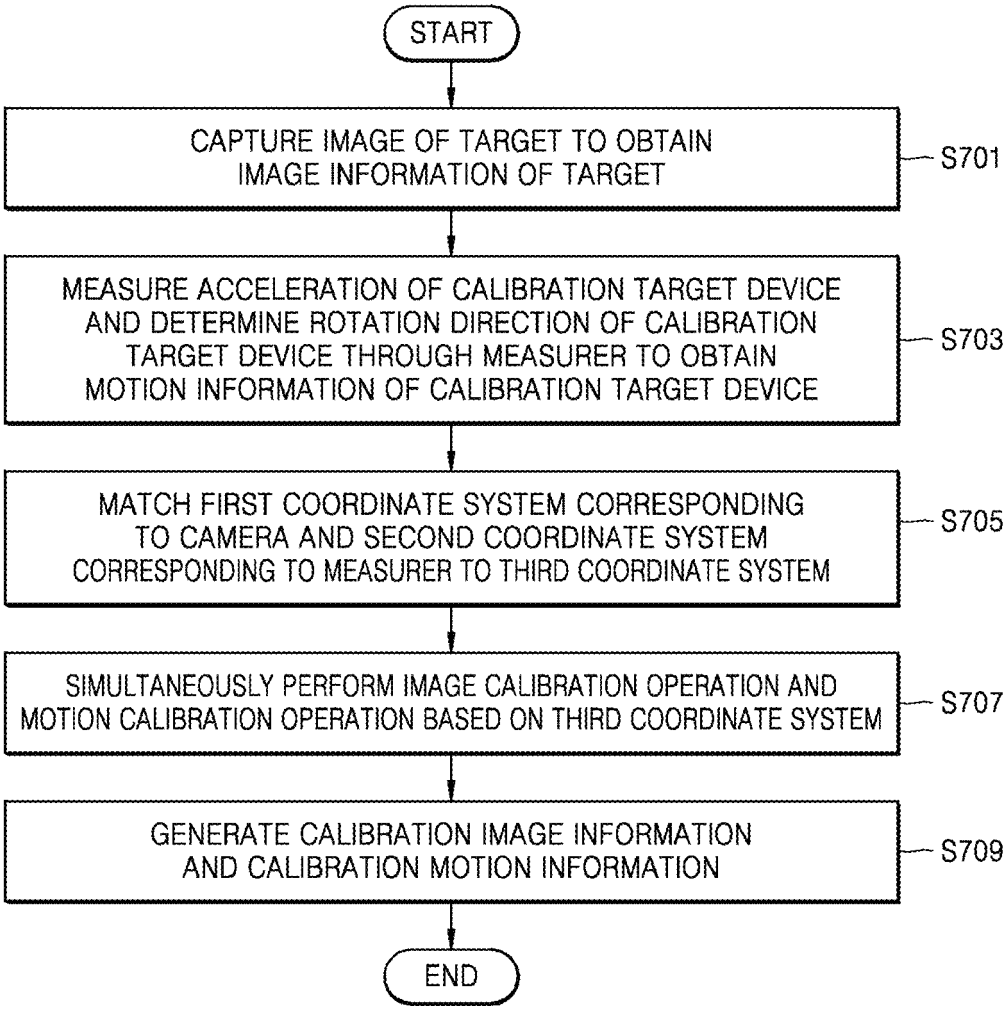

START
CAPTURE IMAGE OF TARGET TO OBTAIN IMAGE INFORMATION OF TARGET
S701
MEASURE ACCELERATION OF CALIBRATION TARGET DEVICE AND DETERMINE ROTATION DIRECTION OF CALIBRATION TARGET DEVICE THROUGH MEASURER TO OBTAIN MOTION INFORMATION OF CALIBRATION TARGET DEVICE
S703
MATCH FIRST COORDINATE SYSTEM CORRESPONDING TO CAMERA AND SECOND COORDINATE SYSTEM CORRESPONDING TO MEASURER TO THIRD COORDINATE SYSTEM
S705
SIMULTANEOUSLY PERFORM IMAGE CALIBRATION OPERATION AND MOTION CALIBRATION OPERATION BASED ON THIRD COORDINATE SYSTEM
S707
GENERATE CALIBRATION IMAGE INFORMATION AND CALIBRATION MOTION INFORMATION
S709
END

FIG. 8

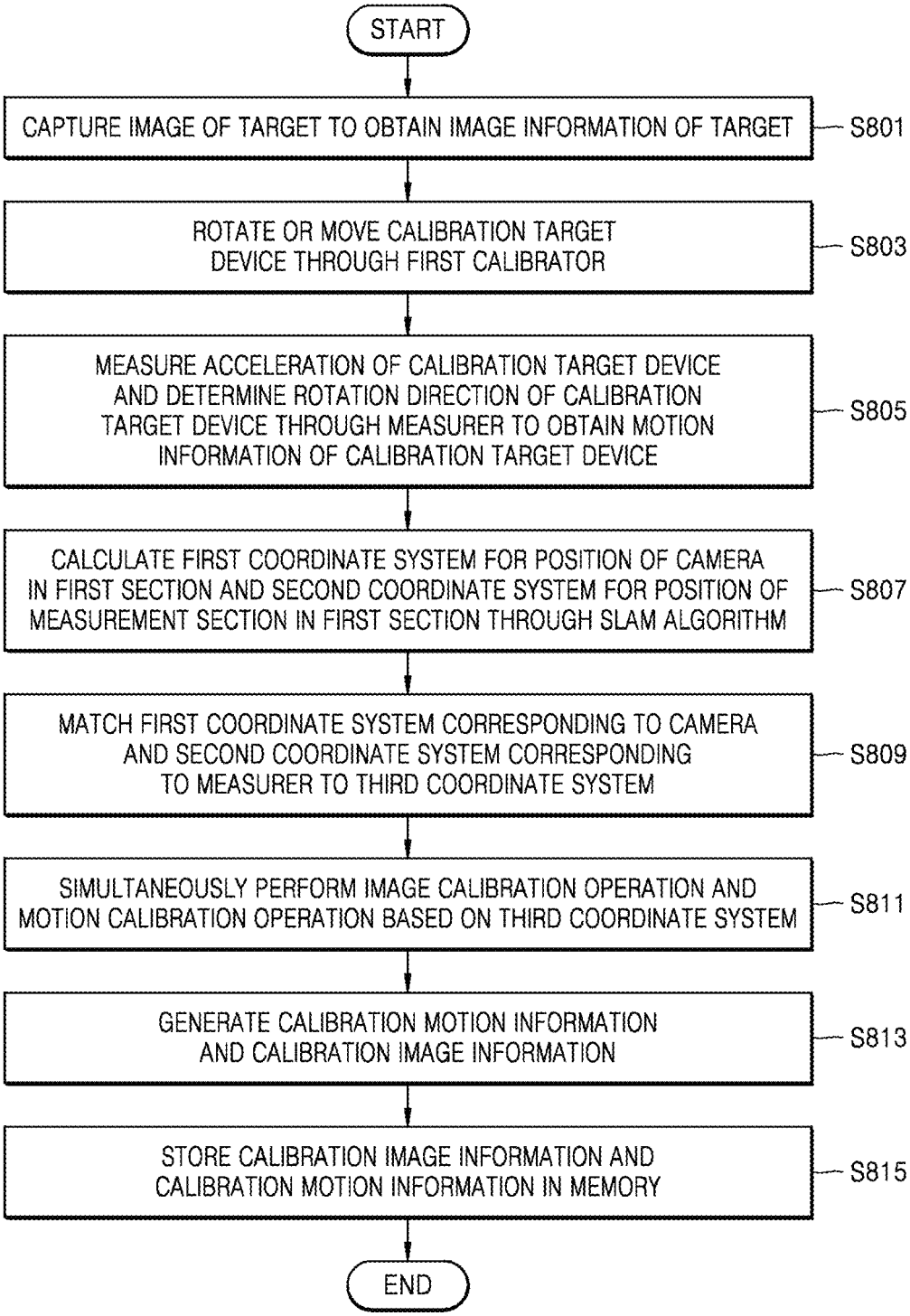
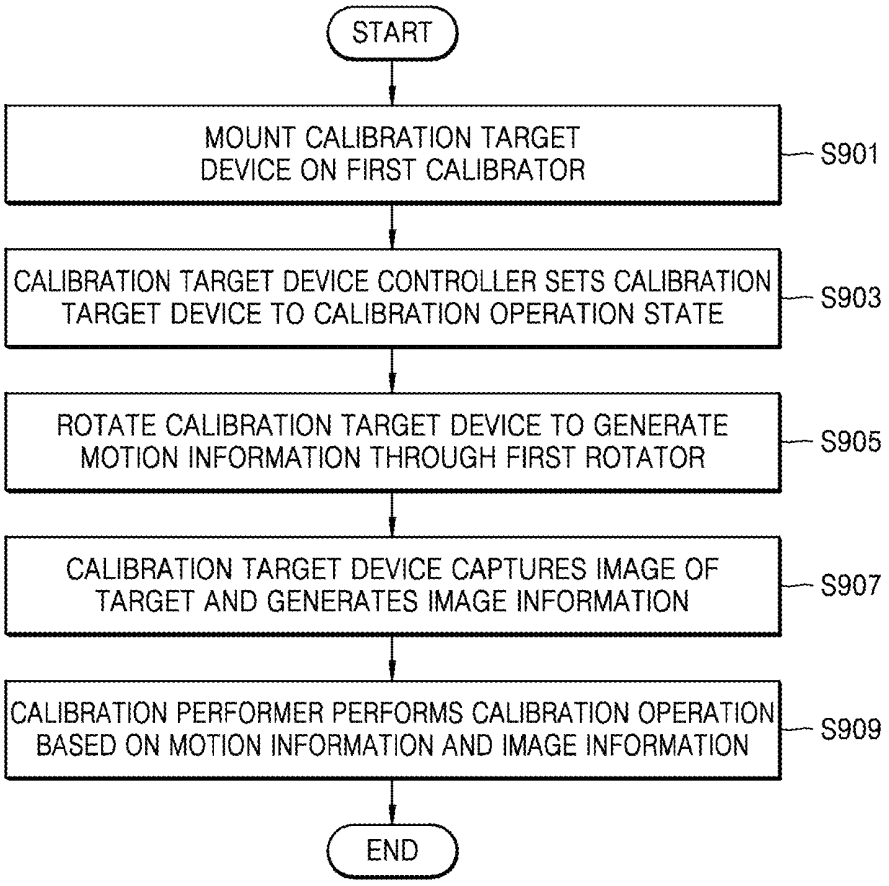

START
CAPTURE IMAGE OF TARGET TO OBTAIN IMAGE INFORMATION OF TARGET
S801
ROTATE OR MOVE CALIBRATION TARGET DEVICE THROUGH FIRST CALIBRATOR
S803
MEASURE ACCELERATION OF CALIBRATION TARGET DEVICE AND DETERMINE ROTATION DIRECTION OF CALIBRATION TARGET DEVICE THROUGH MEASURER TO OBTAIN MOTION INFORMATION OF CALIBRATION TARGET DEVICE
S805
CALCULATE FIRST COORDINATE SYSTEM FOR POSITION OF CAMERA IN FIRST SECTION AND SECOND COORDINATE SYSTEM FOR POSITION OF MEASUREMENT SECTION IN FIRST SECTION THROUGH SLAM ALGORITHM
S807
MATCH FIRST COORDINATE SYSTEM CORRESPONDING TO CAMERA AND SECOND COORDINATE SYSTEM CORRESPONDING TO MEASURER TO THIRD COORDINATE SYSTEM
S809
SIMULTANEOUSLY PERFORM IMAGE CALIBRATION OPERATION AND MOTION CALIBRATION OPERATION BASED ON THIRD COORDINATE SYSTEM
S811
GENERATE CALIBRATION MOTION INFORMATION AND CALIBRATION IMAGE INFORMATION
S813
STORE CALIBRATION IMAGE INFORMATION AND CALIBRATION MOTION INFORMATION IN MEMORY
S815
END

FIG. 9


START
MOUNT CALIBRATION TARGET DEVICE ON FIRST CALIBRATOR
S901
CALIBRATION TARGET DEVICE CONTROLLER SETS CALIBRATION TARGET DEVICE TO CALIBRATION OPERATION STATE
S903
ROTATE CALIBRATION TARGET DEVICE TO GENERATE MOTION INFORMATION THROUGH FIRST ROTATOR
S905
CALIBRATION TARGET DEVICE CAPTURES IMAGE OF TARGET AND GENERATES IMAGE INFORMATION
S907
CALIBRATION PERFORMER PERFORMS CALIBRATION OPERATION BASED ON MOTION INFORMATION AND IMAGE INFORMATION
S909
END

CALIBRATION DEVICE AND METHOD OF OPERATING THE SAME

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application is based on and claims priority under 35 U.S.C. § 119(a) of a Korean patent application number 10-2018-0078930, filed on Jul. 6, 2018, in the Korean Intellectual Property Office, the disclosure of which is incorporated by reference herein in its entirety.

BACKGROUND

1. Field

The disclosure relates to calibration devices and methods of operating the calibration devices.

2. Description of Related Art

With the recent advances in information technology (IT) technology, cameras have evolved from traditional film cameras to digital cameras. Also, in recent years, stereo cameras capable of detecting three-dimensional distances or depths have been developed.

The above information is presented as background information only to assist with an understanding of the disclosure. No determination has been made, and no assertion is made, as to whether any of the above might be applicable as prior art with regard to the disclosure.

SUMMARY

Aspects of the disclosure are to address at least the above-mentioned problems and/or disadvantages and to provide at least the advantages described below. Accordingly, an aspect of the disclosure is to provide calibration devices with an improved operating range and speed and methods of operating the calibration devices.

Additional aspects will be set forth in part in the description which follows and, in part, will be apparent from the description, or may be learned by practice of the presented embodiments.

In accordance with an aspect of the disclosure, a calibration device is provided. The calibration device includes a first calibrator configured to rotate or move a calibration target device including at least one camera configured to capture an image of target and acquire image information of the target and at least one measurer configured to acquire motion information of the calibration target device based on an acceleration of the calibration target device and a rotation direction of the calibration target device and at least one processor configured to match a first coordinate system corresponding to the at least one camera and a second coordinate system corresponding to the at least one measurer to a third coordinate system, simultaneously perform, based on the third coordinate system, an image calibration operation with respect to the image information and a motion calibration operation with respect to the motion information, and generate calibration motion information via the image calibration operation with respect to the image information, and generate calibration motion information via the motion calibration operation with respect to the motion information.

In accordance with another aspect of the disclosure, a calibration method is provided. The calibration method includes capturing an image of target and acquiring image information of the target, acquiring motion information of the calibration target device by measuring an acceleration of the calibration target device and by determining a rotation direction of the calibration target device and matching a first coordinate system corresponding to a camera configured to capture the image of the target and a second coordinate system corresponding to a measurer configured to acquire the motion information of the calibration target device to a third coordinate system simultaneously performing, based on the third coordinate system, an image calibration operation with respect the image information and a motion calibration operation with respect the motion information, generating calibration motion information via the image calibration operation with respect to the image information, and generating calibration motion information via the motion calibration operation with respect to the motion information.

In accordance with another aspect of the disclosure, a computer program product is provided. The computer program product includes a non-transitory computer readable recording medium including a program to perform operations of, capturing a target and acquiring image information of the target, acquiring motion information of the calibration target device by measuring an acceleration of the calibration target device and by determining a rotation direction of the calibration target device and matching a first coordinate system corresponding to a camera configured to capture an image of the target and a second coordinate system corresponding to a measurer configured to acquire the motion information of the calibration target device to a third coordinate system, simultaneously performing, based on the third coordinate system, an image calibration operation with respect to the image information and a motion calibration operation with respect to the motion information, generating calibration motion information via the image calibration operation with respect to the image information, and generating calibration motion information via the motion calibration operation with respect to the motion information.

Other aspects, advantages, and salient features of the disclosure will become apparent to those skilled in the art from the following detailed description, which, taken in conjunction with the annexed drawings, discloses various embodiments of the disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other aspects, features, and advantages of certain embodiments of the disclosure will be more apparent from the following description taken in conjunction with the accompanying drawings, in which:

FIG. **1**A illustrates a first calibrator according to an embodiment of the disclosure;

FIG. **1**B illustrates a calibration device according to an embodiment of the disclosure;

FIG. **2** illustrates a configuration of a computer device according to an embodiment of the disclosure;

FIG. **3**A shows a front view of a calibration target device according to an embodiment of the disclosure;

FIG. **3**B shows a top view of a calibration target device according to an embodiment of the disclosure;

FIG. **4** illustrates examples of a target according to an embodiment of the disclosure;

FIG. **5**A illustrates a method of rotating a calibration target device according to an embodiment of the disclosure;

FIG. **5**B illustrates a method of rotating a calibration target device according to an embodiment of the disclosure;

FIG. **6** shows a calibration operation result according to an embodiment of the disclosure;

FIG. **7** is a flowchart illustrating a calibration operating method according to an embodiment of the disclosure;

FIG. **8** is a flowchart illustrating a calibration operating method according to an embodiment of the disclosure; and

FIG. **9** is a flowchart illustrating a calibration operating method according to an embodiment of the disclosure.

Throughout the drawings, it should be noted that like reference numbers are used to depict the same or similar elements, features, and structures.

DETAILED DESCRIPTION

The following description with reference to the accompanying drawings is provided to assist in a comprehensive understanding of various embodiments of the disclosure as defined by the claims and their equivalents. It includes various specific details to assist in that understanding but these are to be regarded as merely exemplary. Accordingly, those of ordinary skill in the art will recognize that various changes and modifications of the various embodiments described herein can be made without departing from the scope and spirit of the disclosure. In addition, descriptions of well-known functions and constructions may be omitted for clarity and conciseness.

The terms and words used in the following description and claims are not limited to the bibliographical meanings, but, are merely used by the inventor to enable a clear and consistent understanding of the disclosure. Accordingly, it should be apparent to those skilled in the art that the following description of various embodiments of the disclosure is provided for illustration purpose only and not for the purpose of limiting the disclosure as defined by the appended claims and their equivalents.

It is to be understood that the singular forms "a," "an," and "the" include plural referents unless the context clearly dictates otherwise. Thus, for example, reference to "a component surface" includes reference to one or more of such surfaces.

It will be understood that, although the terms first, second, etc. may be used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another.

The terminology used herein is for the purpose of describing embodiments only and is not intended to be limiting of embodiments. As used herein, the singular forms "a," "an," and "the," are intended to include the plural forms as well, unless the context clearly indicates otherwise. Throughout the specification, it will be understood that when an element is referred to as being "connected" to another element, it may be "directly connected" to the other element or "electrically connected" to the other element with intervening elements therebetween. It will be further understood that when a part "includes" or "comprises" an element, unless otherwise defined, the part may further include other elements, not excluding the other elements.

The use of the terms "a" and "an" and "the" and similar referents in the context of describing the disclosure are to be construed to cover both the singular and the plural. Also, the steps of all methods described herein may be performed in any suitable order unless otherwise indicated herein or otherwise clearly contradicted by context. The disclosure is not limited to the described order of the steps.

In the disclosure, the expression 'some embodiments' or 'an embodiment' do not always indicate the same embodiment.

The disclosure may be described in terms of functional block components and various processing steps. Such functional blocks may be realized by any number of hardware and/or software components configured to perform the specified functions. For example, the functional blocks of the disclosure may be implemented by one or more microprocessors or circuit components for a predetermined function. Also, for example, the functional blocks of the disclosure may be implemented with any programming or scripting language. Functional aspects may be implemented in algorithms that are executed on one or more processors. Furthermore, the disclosure could employ any number of techniques according to the related art for electronics configuration, signal processing and/or control, data processing and the like. The words "mechanism", "element", "means", and "configuration" are used broadly and are not limited to mechanical or physical embodiments, but may include software routines in conjunction with processors, etc.

Furthermore, the connecting lines, or connectors shown in the various figures presented are intended to represent functional relationships and/or physical or logical couplings between the various elements. It should be noted that many alternative or additional functional relationships, physical connections or logical connections may be present in a practical device.

A stereo camera may include a plurality of camera modules and synthesize image data imaged received from the plurality of camera modules and provides a user with an image having a sense of depth. An inertial measurement unit (IMU) is partially embedded in equipment such as unmanned aerial vehicles (drone), airplanes, satellites, etc. and is an integrated unit for measuring the acceleration and rotation rate, i.e., inertia, by using acceleration sensors and gyroscope sensors and also measure a motion in a three-dimensional space. The IMU may also measure the magnetic field for azimuths. Recently, when it is not possible to receive global positioning system (GPS) signals in tunnels, buildings, or spaces where electromagnetic interference is present, a technique for estimating a location based on information of the IMU is useful in a system such as a navigation system (GPS). In modern times, electronic devices basically include cameras and IMUs. To operate the stereo camera and IMU, it is necessary to calibrate each of them.

Hereinafter, the disclosure will be described in detail by explaining preferred embodiments of the disclosure with reference to the attached drawings.

FIG. **1**A illustrates a first calibrator according to an embodiment of the disclosure.

Referring to FIG. **1**A, a first calibrator **110** according to an embodiment may include a calibration target device **111**, a first supporter **112**, and a first rotator **113**.

The calibration target device **111** according to an embodiment may include at least one camera (not shown) capturing a target (not shown) to obtain image information of the target. The calibration target device **111** may also include at least one measurer (not shown) acquiring motion information of the calibration target device **111** based on at least one sensor of an acceleration sensor (not shown) and a gyroscope sensor (not shown). The calibration target device **111** may include at least one of a camera and an IMU.

A measurer according to an embodiment may mean an IMU. That is, according to an embodiment, the calibration target device **111** may include at least one IMU.

An acceleration sensor measures an acceleration of a moving object or an intensity of an impact between a moving object and an obstacle. An acceleration sensor

according to an embodiment may measure an acceleration of the calibration target device **111**. A gyroscope sensor is a device measuring the rotational repulsive force generated when an object rotates and converting the rotational repulsive force into an electric signal. A gyroscope sensor according to an embodiment may measure a direction of rotation of the calibration target device **111**.

The calibration target device **111** may also include a sensor capable of measuring a velocity, acceleration, azimuth, tilt, gravity, rotation, geomagnetic field, and magnetic field, including the acceleration sensor and the gyroscope sensor. That is, the motion information described in an embodiment may mean information about velocity, acceleration, azimuth, tilt, gravity, rotation, geomagnetic field, and magnetic field.

The calibration target device **111** according to an embodiment may include a mobile phone, a tablet, a smart watch, a surround view device, a TV, a computer, a notebook, a virtual reality device, an augmented reality device, a mixed reality device, a video security device, a navigation device, and the like. It should be noted, however, that according to a type of the calibration target device **111**, the calibration target device **111** of a specific type may not include the IMU.

When the camera included in the calibration target device **111** captures an image of the target before calibration is performed, the captured image may basically include distorted information. Therefore, a calibration device **100** according to an embodiment may be used to calibrate the calibration target device **111** including the camera that captured the image with respect to distorted information. The calibration target device **111** on which calibration is performed may obtain image information more similar to an actual image of the captured image of the target than before calibration is performed.

According to a similar principle, when motion occurs in the calibration target device **111** before calibration on the IMU is performed, motion information generated by the IMU may basically include distorted information. Accordingly, the calibration device **100** according to an embodiment may calibrate the IMU included in the calibration target device **111** with respect to the generated motion information. The calibration target device **111** on which calibration is performed may obtain more accurate motion information than before calibration is performed.

The first supporter **112** according to an embodiment may support the calibration target device **111** to be positioned within the calibration device **100**. As shown in FIG. **1**A, the first supporter **112** may support more than one calibration target device **111**. For example, as shown in FIG. **1**A, the first supporter **112** may simultaneously support the twelve calibration target devices. By simultaneously supporting the plurality of calibration target devices, the calibration device **100** according to an embodiment may simultaneously perform a calibration operation on the plurality of calibration target devices. Accordingly, when there are the plurality of calibration target devices on which calibration is to be performed, the calibration operation may be simultaneously performed, thereby reducing the time for performing the entire calibration operation.

According to an embodiment, the first calibrator **110** may include the first rotator **113** that allows the calibration target device **111** to generate the motion information.

The first rotator **113** (or a rotation mover) according to an embodiment may perform a movement or rotation operation such that the calibration target device **111** may generate the motion information. According to an embodiment, the first rotator **113** may rotate in a 360-degree direction to move the calibration target device **111**. According to an embodiment, the first rotator **113** may repeatedly move the calibration target device **111** by a predetermined angle. The first rotator **113** according to an embodiment may move the calibration target device **111** such that the acceleration sensor and the gyroscope sensor may measure information about a velocity, azimuth, tilt, magnetic field, etc. The IMU may acquire the motion information of the calibration target device **111** in accordance with the movement or rotation operation of the first rotator **113** according to an embodiment. The first rotator **113** according to an embodiment may be a device including all physical devices capable of moving the calibration target device **111**.

According to an embodiment, the first calibrator **110** may include a processor (not shown). The processor according to an embodiment may match a first coordinate system corresponding to the camera and a second coordinate system corresponding to the measurer to a third coordinate system and, based on the third coordinate system, simultaneously perform an image calibration operation on the image information and a motion calibration operation on the motion information, thereby generating calibration motion information, which is a result obtained by calibrating the image information and calibration motion information, which is a result obtained by calibrating the motion information. The operation of the processor will be described in more detail later with reference to FIG. **2**.

FIG. **1**B illustrates a calibration device according to an embodiment of the disclosure.

Referring to FIG. **1**B, the calibration device **100** according to an embodiment may include the first calibrator **110** described with reference to FIG. **1**A and further include a second calibrator **130** and a computer device **200**. That is, the calibration device **100** according to an embodiment may include the first calibrator **110**, the second calibrator **130**, and the computer device **200**.

The first calibrator **110** according to an embodiment may further include the first rotator **113**, a first mover **115**, and a second mover **117**. That is, the first calibrator **110** according to an embodiment may include the calibration target device **111**, the first supporter **112**, the first rotator **113**, the first mover **115**, and the second mover **117**.

The first calibrator **110** according to an embodiment may include the first mover **115** capable of moving the calibration target device **111** in a Z axis direction. The first mover **115** may move the calibration target device **111** in the Z axis direction to a position at a distance suitable for a camera of the calibration target device **111** to capture an image of a target **131**.

The first calibrator **110** according to an embodiment may include the second mover **117** capable of moving the calibration target device **111** in a Y axis direction. The second mover **117** may adjust a distance between the calibration target device **111** and the target **131** by moving the calibration target device **111** in the Y axis direction.

The second calibrator **130** according to an embodiment may include the target **131**, a second supporter **132**, a second rotator **133**, a third mover **135**, a fourth mover **137**, a fifth mover **138**, and a sixth mover **139**.

The target **131** according to an embodiment may be a display device for calibrating the camera included in the calibration target device **111**. The target **131** may be an electronic display device or a plate directly replaceable by a user.

The calibration target device **111** according to an embodiment may acquire image information of the target **131** by capturing the target **131**.

The second supporter **132** according to an embodiment may support the target **131**. In FIG. **1**B, the second supporter **132** supports the target **131**, but according to another embodiment, more than one second supporter **132** may be provided to respectively support more than one target **131**.

The second rotator **133** according to an embodiment may be connected to the second supporter **132** to rotate the second supporter **132**, thereby rotating the target **131**. The second rotator **133** may rotate the target **131** to position the target **131** at a position suitable for the camera of the calibration target device **111** to capture the image of the target **131**.

The third mover **135** according to an embodiment may be connected to the second rotator **133** or the second supporter **132**. The third mover **135** may move in the Z-axis direction such that the target **131** may move in the Z-axis direction. The third mover **135** may move the target **131** in the Z axis direction to position the target **131** at the position suitable for the camera of the calibration target device **111** to capture the image of the target **131**.

The fourth mover **137** according to an embodiment may be connected to the third mover **135**. The fourth mover **137** may move the third mover **135** in the Y-axis direction such that the target **131** may move in the Y-axis direction. The fourth mover **137** may move the target **131** in the Y axis direction to position the target **131** at the position suitable for the camera of the calibration target device **111** to capture the image of the target **131**.

The fifth mover **138** according to an embodiment may be connected to the fourth mover **137**. The fifth mover **138** may move the fourth mover **137** in the X axis direction such that the target **131** may in the X axis direction. The fifth mover **138** may move the target **131** in the X axis direction to position the target **131** at the position suitable for the camera of the calibration target device **111** to capture the image of the target **131**.

The sixth mover **139** according to an embodiment may be connected to the fifth mover **138**. The sixth mover **139** may move the fifth mover **138** in the Y-axis direction such that the target **131** may move in the Y-axis direction. The sixth mover **139** may move the target **131** in the Y axis direction to position the target **131** at the position suitable for the camera of the calibration target device **111** to capture the image of the target **131**.

The coupling relationship among the third mover **135**, the fourth mover **137**, the fifth mover **138**, and the sixth mover **139** according to an embodiment is not limited to the coupling relationship shown in FIG. **1**B. For example, although the third mover **135** is directly coupled to the fourth mover **137** in FIG. **1**B, in another embodiment, the third mover **135** may be directly connected to the fifth mover **138**. In another embodiment, the third mover **135**, the fourth mover **137**, the fifth mover **138**, and the sixth mover **139** may move in the X, Y, and Z axes as one integrated device.

The computer device **200** according to an embodiment may be connected to the first calibrator **110** and the second calibrator **130** through a communication path **104**.

The communication path **104** may be a network of various kinds. For example, the communication path may include wireless communication, wired communication, optical communication, ultrasonic communication, or a combination thereof. The communication path **104** may also include satellite communication, mobile communication, Bluetooth, WiFi, infrared data association standard (IrDA), WiMAX, and the like. Ethernet, digital subscriber line (DSL), fiber to the home (FTTH), and existing general telephone services are also examples of wired communication that may be included in the communication path.

According to an embodiment, the computer device **200** may be included in the first calibrator **110** integrally with the first calibrator **110** or may be included in the second calibrator **130** integrally with the second calibrator **130**. According to an embodiment, the first calibrator **110** may include the computer device **200**, and the first calibrator **110** may include a processor.

FIG. **2** illustrates a configuration of a computer device according to an embodiment of the disclosure.

Referring to FIG. **2**, the computer device **200** according to an embodiment may include a processor **210**, a calibration performer **215**, a first calibration device controller **220**, a second calibration device controller **230**, a calibration target device controller **235**, a display **240**, a memory **250**, an input/output device controller **260**, and a power supplier **270**.

The processor **210** according to an embodiment may control each of the calibration performer **215**, the first calibration device controller **220**, the second calibration device controller **230**, the calibration target device controller **235**, the display **240**, the memory **250**, the input/output device controller **260**, and the power supplier **270**. The processor **210** may include one or more of a central processing unit, an application processor, or a communication processor (CP). The processor **120** may perform, for example, operations or data processing relating to control and/or communication of at least one other component of an electronic device.

The processor **210** according to an embodiment may include at least one of the calibration performer **215**, the first calibration device controller **220**, the second calibration device controller **230**, and the calibration target device controller **235**. That is, the processor **210** may perform an operation of at least one of the calibration performer **215**, the first calibration device controller **220**, the second calibration device controller **230**, and the calibration target device controller **235**.

The processor **210** according to an embodiment may generate calibration motion information based on the first calibrator **110** and generate calibration image information based on the second calibrator **130**.

The calibration performer **215** according to an embodiment may generate at least one of the calibration motion information and the calibration image information, based on at least one of image information of the target **131** received from the first calibrator **110** or the first calibration device controller **220** and motion information of a calibration target device.

The calibration performer **215** according to an embodiment may match a first coordinate system corresponding to a camera and a second coordinate system corresponding to an IMU to a third coordinate system, which is a standard coordinate system.

The first coordinate system corresponding to the camera may be a coordinate system set with reference to the camera. In an embodiment, when the center of a lens included in the camera is the origin of the coordinate system, the coordinate system may be set such that a front optical axis direction of the camera is a Z axis, a downward direction of the camera is a Y axis, and a right direction of the camera is an X axis.

The second coordinate system corresponding to the IMU may be a coordinate system set with reference to the IMU. In an embodiment, when the center of the IMU is the origin of the coordinate system, the coordinate system may be set such that a front direction of the center of the IMU is the Z

axis, a downward direction of the center of the IMU is the Y axis, and a right direction of the center of the IMU is the X axis.

The third coordinate system, which is the standard coordinate system, may be a coordinate system that unifies the first and second coordinate systems that are different from each other. By unifying the first and second coordinate systems which are different from each other into the third coordinate system, the calibration performer **215** according to an embodiment may simultaneously perform an image calibration operation and a motion calibration operation. In other words, the image calibration operation on the first coordinate system and the motion calibration operation on the second coordinate system are not separately performed, and the first and second coordinate systems may be matched to the third coordinate system, and then the image calibration operation and the motion calibration operation may be simultaneously performed on the third coordinate system. Accordingly, the calibration performer **215** according to an embodiment may quickly perform a calibration operation.

The calibration performer **215** according to an embodiment may match the first coordinate system corresponding to the camera and the second coordinate system corresponding to the IMU, which are determined through a simultaneous localization and mapping (SLAM) algorithm, to the third coordinate system, which is the standard coordinate system.

The SLAM algorithm may be an algorithm that simultaneously performs position measurement and mapping. The SLAM algorithm may include a localization process of self-locating and a mapping process of registering an image.

The calibration performer **215** according to an embodiment may calculate a position of the camera in a first section through the SLAM algorithm Accordingly, the calibration performer **215** may calculate a trajectory of a movement of the camera during the calibration operation. The first coordinate system corresponding to the camera may be a coordinate system with respect to the position of the camera in the first section.

Also, the calibration performer **215** according to an embodiment may calculate a position of the IMU in the first section through the SLAM algorithm Accordingly, the calibration performer **215** may calculate a trajectory of a movement of the IMU during the calibration operation. The second coordinate system corresponding to the IMU may be a coordinate system with respect to the position of the IMU in the first section.

The first section may be a time interval between first and second points of time that are 2 points of time. Each of the first and second points of time may be a predetermined point of time necessary for performing the calibration operation. For example, a point of time when the camera captures the image of the target **131** is the first point of time, and a point of time after movement occurs in the calibration target device through the first rotator **113** may be the second point of time. However, the first and second points of time are merely examples, and each of the first and second points of time may be any point of time designated among time for performing the calibration operation. Each of the first and second points of time may be a point of time designated by the user or a predetermined point of time.

The trajectory of movement of the camera may include position and orientation information of the camera between fixed points of time. The calibration performer **215** may calculate information about movement occurred in a camera coordinate system during a period between the first and second points of time from a direction of the camera with respect to each of the first and second points of time. The trajectory of the IMU may include position and orientation information of the IMU at each fixed point of time. The calibration performer **215** may calculate information about movement occurred in an IMU coordinate system during the period between the first and second points of time from a direction of the IMU with respect to any two points of time.

The calibration performer **215** according to an embodiment may calculate a transformation matrix for transforming the first coordinate system and the second coordinate system into the third coordinate system with a minimum error to calibrate the camera and the IMU. The transformation matrix may be a predetermined matrix value used in performing coordinate transformation, which is an operation of changing coordinates of one coordinate system to coordinates of another coordinate system.

In calculating the transformation matrix, the Gauss-Newton method for solving the nonlinear least squares problem may be used. The Gauss-Newton method may be an extension of the Newton's method, which is a method used to approximate the solution of an equation, to the form of simultaneous equations.

The first calibration device controller **220** according to an embodiment may control each of the calibration target device **111**, the first supporter **112**, the first rotator **113**, the first mover **115** and the second mover **117** that are included in the first calibrator **110**.

The first calibration device controller **220** may control the calibration target device **111** to control the camera and the IMU included in the calibration target device **111**. The first calibration device controller **220** may acquire image information of the target **131** by controlling the camera and capturing the target **131**.

The first calibration device controller **220** may control the first supporter **112** such that the first supporter **112** may support the calibration target device **111**. The first calibration device controller **220** may control the first rotator **113** such that the calibration target device **111** may generate motion information. The first calibration device controller **220** may control the first mover **115** to move the calibration target device **111** in a Z axis direction. The first calibration device controller **220** may control the second mover **117** to move the calibration target device **111** in a Y axis direction.

The second calibration device controller **230** according to an embodiment may control each of the target **131**, the second supporter **132**, the second rotator **133**, the fourth mover **137**, the fifth mover **138**, and the sixth mover **139** that are included in the second calibrator **130**.

The second calibration device controller **230** may display an image suitable for the calibration target device **111** to acquire image information of the target **131** on the target **131**. However, this is only an embodiment, and as described above, the target **131** may be a plate directly replaceable by a user.

The second calibration device controller **230** may control the second supporter **132** to fix the target **131**. The second calibration device controller **230** may control the second rotator **133** to rotate the target **131**. The second calibration device controller **230** may control the third mover **135** to move the target **131** in the Z-axis direction. The second calibration device controller **230** may control the fourth mover **137** to move the target **131** in the Y axis direction. The second calibration device controller **230** may control the fifth mover **138** to move the target **131** in the X axis direction. The second calibration device controller **230** may control the sixth mover **139** to move the target **131** in the Y axis direction.

The calibration target device controller **235** according to an embodiment may control the calibration target device **111** supported by the first supporter **112** of the first calibrator **110**. According to an embodiment, the calibration target device controller **235** may set the calibration target device **111** to a calibration operation state. The calibration operation state may mean a preparation state of at least one of hardware and software of the calibration target device **111** for performing at least one calibration operation among the image calibration operation and the motion calibration operation. In an embodiment, the calibration operation state may mean a state in which the software for the calibration operation is executed.

The display **240** according to an embodiment may include a set of devices for displaying a state of the computer device **200**. The display **240** may be a device attached to the computer device **200** or a device present outside the computer device **200**. The processor **210** may control the display **240** to display the operation state of the computer device **200**.

The memory **250** according to an embodiment may include volatile and/or non-volatile memory. The memory **250** may store instructions or data related to, for example, at least one other component of an electronic device. According to an embodiment, the memory **250** may store software and/or programs. According to an embodiment, the memory may store the calibration image information and the calibration motion information.

The memory **250** may include, for example, an internal memory or an external memory. The internal memory may include at least one of, for example, a volatile memory (e.g., dynamic random-access memory (DRAM), static RAM (SRAM), or synchronous DRAM (SDRAM), a non-volatile memory (e.g., an one time programmable (OTPROM), a programmable ROM (PROM), an erasable PROM (EPROM), an electronically EPROM (EEPROM), a mask ROM, a flash ROM, a flash memory, a hard drive, a solid state drive (SSD). The external memory may include a flash drive, for example, a compact flash (CF), a secure digital (SD), Micro-SD, Mini-SD, a multi-media card (MMC), a memory stick, etc. The external memory may be functionally or physically connected to the electronic device through various interfaces.

The input/output device controller **260** according to an embodiment may be a device that encompasses devices for the user to interact with the computer device **200**. For example, the input/output device controller **260** may be a device that controls an input/output device such as a mouse, a keyboard, a monitor, etc. The processor **210** may control the input/output device controller **260** to enable the user to interact with the computer device **200**.

The power supplier **270** according to an embodiment may supply power necessary for the operation of the computer device **200**. The processor **210** may control the power supplied by the power supplier **270**.

However, in the above-described embodiment, it is assumed that the computer device **200** exists as a separate device from the first calibrator **110** and the second calibrator **130**, but the computer device **200** may be included in at least one of the first calibrator **110** and the second calibrator **130** in another embodiment. For example, the first calibrator **110** may include the computer device **200**. That is, the first calibrator **110** may include the processor **210** included in the computer device **200**. Therefore, the processor **210** included in the first calibrator **110** may match the first coordinate system corresponding to the camera and the second coordinate system corresponding to the measurer to the third coordinate system, and based on the third coordinate system, simultaneously perform the image calibration operation on the image information and the motion calibration operation on the motion information, thereby generating calibration motion information which is a result obtained by calibrating the image information and calibration motion information which is a result obtained by calibrating the motion information.

FIG. **3**A shows a front view of a calibration target device according to an embodiment of the disclosure.

Referring to FIG. **3**A, the calibration target device **111** according to an embodiment may include a plurality of cameras **311** and **313** and a single IMU **315**. However, the number, sizes, and positions of each of cameras and IMUs may vary according to a type of the calibration target device **111**. For example, when the calibration target device **111** is a mobile phone or tablet, the calibration target device **111** may include a single camera and a single IMU. When the calibration target device **111** is a smart watch or a navigation device, the calibration target device **111** may include a plurality of IMUs having a relatively large sensor.

As shown in FIG. **3**A, the plurality of cameras **311** and **313** and the single IMU **315** may be different from each other in the size and position. For example, the single IMU **315** may be larger than the plurality of cameras **311** and **313**. This may mean that there is a difference between the center of the single IMU **315** and the center of the plurality of cameras **311** and **313**.

Accordingly, the calibration performer **215** according to an embodiment may match a first coordinate system corresponding to a camera and a second coordinate system corresponding to an IMU to a third coordinate system which is a standard coordinate system through a SLAM algorithm

The calibration performer **215** according to an embodiment may perform a calibration operation based on the third coordinate system to quickly perform image calibration and motion calibration.

FIG. **3**B shows a top view of a calibration target device according to an embodiment of the disclosure.

Referring to FIG. **3**B, a plurality of cameras **321** and **323** and a single IMU **325** may be different from each other in the size and position. For example, the single IMU **325** may be in a somewhat higher position in a plan view than the plurality of cameras **321** and **323**. This may mean that the center of the single IMU **325** is located at a position higher than the center of the plurality of cameras **321** and **323**.

Accordingly, the calibration performer **215** according to an embodiment may match a first coordinate system corresponding to a camera and a second coordinate system corresponding to an IMU to a third coordinate system which is a standard coordinate system through a SLAM algorithm

The calibration performer **215** according to an embodiment may perform a calibration operation based on the third coordinate system to quickly perform image calibration and motion calibration.

FIG. **4** illustrates examples of a target according to an embodiment of the disclosure.

Referring to FIG. **4**, the target **131** according to an embodiment may be a display device for calibrating a camera included in the calibration target device **111**. The target **131** may be an electronic display device or a plate directly replaceable by a user.

The target **131** according to an embodiment may be an image in which a certain pattern is repeated or an image that is easy to measure a distortion state of the camera.

For example, as shown in FIG. **4**, a first pattern **401** shows that circular patterns of predetermined sizes are regularly

arranged at regular intervals, a second pattern **403** shows that circular patterns of small sizes are irregularly arranged, and a third pattern **405** shows that white and black patterns are irregularly mixed without a specific figure.

The first pattern **401**, the second pattern **403** and the third pattern **405** may be used alone or in combination for an image calibration operation of the calibration target device **111**.

FIG. **5**A illustrates a method of rotating a calibration target device according to an embodiment of the disclosure.

Referring to FIG. **5**A, the first calibrator **110** may include the first rotator **113** that allows the calibration target device **111** to generate motion information.

According to an embodiment, the calibration target device **111** may include at least one IMU acquiring the motion information of the calibration target device **111** based on at least one sensor of an acceleration sensor and a gyroscope sensor.

The first rotator **113** according to an embodiment may perform movement and rotation operations such that the calibration target device **111** may generate the motion information. According to an embodiment, the first rotator **113** may repeatedly move the calibration target device **111** at a predetermined angle. As shown in FIG. **5**A, the first rotator **113** may rotate in a 180 degree direction to move the calibration target device **111**. The first rotator **113** may rotate the calibration target device **111** six times in the 180 degree direction including the number of rotation **1 501**, the number of rotation **2 502**, the number of rotation **3 503**, the number of rotation **4 504**, the number of rotation **5 505**, and the number of rotation **6 506**.

The first rotator **113** according to an embodiment may move the calibration target device **111** such that an acceleration sensor and a gyroscope sensor may measure information about velocity, azimuth, tilt, and magnetic field.

FIG. **5**B illustrates another embodiment of a method of rotating a calibration target device according to an embodiment of the disclosure.

Referring to FIG. **5**B, the first rotator **113** may repeatedly move the calibration target device **111** at a predetermined angle. As shown in FIG. **5**B, the first rotator **113** may move the calibration target device **111** in a 90 or 180 degree direction.

FIG. **6** shows a calibration operation result according to an embodiment of the disclosure.

Referring to FIG. **6**, before calibration is applied **601**, the image of the target **131** captured by the calibration target device **111** may be somewhat distorted. Specifically, while the target **131** has an actually flat shape, the distorted image of the target **131** may have a bent upper portion.

After calibration is applied **603** to the calibration target device **111** according to an embodiment, the image of the target **131** captured by the calibration target device **111** may be similar to the actual shape of the target **131**.

FIG. **7** is a flowchart illustrating a calibration operating method according to an embodiment of the disclosure.

Referring to FIG. **7**, in operation S**701**, a camera according to an embodiment may capture an image of the target **131** to obtain image information of the target **131**.

In operation S**703**, the processor **210** according to an embodiment may measure the acceleration of the calibration target device **111** and the rotation direction of the calibration target device **111** through a measurer to obtain motion information of the calibration target device **111**.

In operation S**705**, the processor **210** according to an embodiment may match a first coordinate system corresponding to the camera and a second coordinate system corresponding to the measurer to a third coordinate system.

In operation S**707**, the processor **210** according to an embodiment may simultaneously perform an image calibration operation and a motion calibration operation based on the third coordinate system.

In operation S**709**, the processor **210** according to an embodiment may generate calibration image information, which is a result obtained by calibrating the image information and calibration motion information, which is a result obtained by calibrating the motion information.

FIG. **8** is a flowchart illustrating a calibration operating method according to another embodiment of the disclosure.

Referring to FIG. **8**, in operation S**801**, a camera according to an embodiment may capture the image of the target **131** to obtain image information of the target **131**.

In operation S**803**, the processor **210** according to an embodiment may rotate or move the calibration target device **111** through the first calibrator **110**.

In operation S**805**, the processor **210** according to an embodiment may measure the acceleration of the calibration target device **111** and the rotation direction of the calibration target device **111** through a measurer to obtain motion information of the calibration target device **111**.

In operation S**807**, the processor **210** according to an embodiment may calculate a first coordinate system for the position of the camera in the first section and a second coordinate system for the position of the measurement section in the first section through the SLAM algorithm

In operation S**809**, the processor **210** according to an embodiment may match a first coordinate system corresponding to the camera and a second coordinate system corresponding to the measurer to a third coordinate system.

In operation S**811**, the processor **210** according to an embodiment may simultaneously perform an image calibration operation and a motion calibration operation based on the third coordinate system.

In operation S**813**, the processor **210** according to an embodiment may generate calibration motion information, which is a result obtained by calibrating the image information, and calibration image information, which is a result obtained by calibrating the motion information.

In operation S**815**, the processor **210** according to an embodiment may store the calibration image information and the calibration motion information in the memory **250**.

FIG. **9** is a flowchart illustrating a calibration operating method according to another embodiment of the disclosure.

Referring to FIG. **9**, in operation S**901**, a user may mount the calibration target device **111** on the first calibrator **110**. Specifically, the first supporter **112** according to an embodiment may support the calibration target device **111** to be positioned inside the calibration device **100**. Therefore, the user may mount the calibration target device **111** to the first calibrator **110** through the first supporter **112**.

In operation S**903**, the calibration target device controller **235** according to an embodiment may set the calibration target device **111** to a calibration operation state. The calibration operation state may mean a hardware/software preparation state of the calibration target device **111** for performing at least one calibration operation among an image calibration operation and a motion calibration operation. In an embodiment, the calibration operation state may mean a state in which software for the calibration operation is executed.

In operation S**905**, the first calibration device controller **220** according to an embodiment may rotate the calibration target device **111** to generate motion information. The first

rotator **113** according to an embodiment may perform movement and rotation operations such that the calibration target device **111** may generate motion information.

In operation S**907**, the calibration target device controller **235** according to an embodiment may control the calibration target device **111** to capture the image of the target **131** and generate image information.

In operation S**909**, the calibration performer **215** according to an embodiment may perform the calibration operation on the calibration target device **111** based on at least one of the motion information and the image information received from the first calibrator **110**.

It should be understood that embodiments described herein should be considered in a descriptive sense only and not for purposes of limitation. Descriptions of features or aspects within each embodiment should typically be considered as available for other similar features or aspects in other embodiments.

While one or more embodiments have been described with reference to the figures, it will be understood by those of ordinary skill in the art that various changes in form and details may be made therein without departing from the spirit and scope as defined by the following claims.

Those of ordinary skill in the art would understand the block diagrams disclosed in the disclosure as conceptual diagrams of circuits for realizing the principles of the disclosure Similarly, it would be apparent to those of ordinary skill in the art that arbitrary flow charts, flow diagrams, state transition diagrams, pseudo code, and the like denote various processes that may be substantially stored in a computer-readable recording medium and that may be performed by a computer or a processor, regardless of whether the computer or the processor are explicitly illustrated or not. Thus, the embodiments of the disclosure described above may be embodied as a computer program. The computer program may be stored in a computer-readable recording medium, and executed using a general digital computer. Examples of the computer-readable medium are a magnetic recording medium (a ROM, a floppy disc, a hard disc, etc.), and an optical recording medium (a compact disc (CD)-ROM, a digital versatile disc (DVD), etc.).

The functions of various elements illustrated in the drawings may be related to appropriate software, and be provided via not only hardware capable of executing the software but also exclusive hardware. These functions may also be provided via a single exclusive processor, a single shared processor, or a plurality of individual processors, some of which may be shared. Also, explicit use of the term 'processor' or 'controller' is not limited to exclusively using hardware capable of executing software, and may implicitly include hardware such as a digital signal processor (DSP), and a ROM, a RAM, or a non-volatile storage medium for storing software.

In the claims of the present specification, an element suggested as an element for performing a specific operation includes any arbitrary methods of performing the specific operation. Examples of this element may include a combination of circuit elements capable of performing the specific operation, or software having an arbitrary form, e.g., firmware or microcode, which is combined with an appropriate circuit for executing software for performing the specific operation.

In the disclosure, the expression 'an embodiment' of the principles of the disclosure and various modifications of this expression mean that specific features, structure, and characteristics related to this embodiment are included in at least an embodiment of the principles of the disclosure. Thus, the expression 'an embodiment' and arbitrary other modifications thereof disclosed in the disclosure do not always indicate the same embodiment.

In the disclosure, the expression 'at least one of' of 'at least one of A and B' is used to inclusively mean that only the first option (A) is selected, only the second option (B) is selected, or both the first and second operations (A and B) are selected. In addition, the expression 'at least one of A, B, and C' is used to inclusively mean that only the first option (A) is selected, only the second option (B) is selected, only the third option (C) is selected, only the first and second options (A and B) are selected, only the second and third options (B and C) are selected, only the first and third (A and C) are selected or all the three options (A, B, and C) are selected. When more than three items are listed in relation to this expression, the meaning thereof would be apparent to those of ordinary skill in the art.

The embodiments of the disclosure have been described above.

While the disclosure has been shown and described with reference to various embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the disclosure as defined by the appended claims and their equivalents.

What is claimed is:

1. A calibration device comprising:

a first calibrator configured to rotate or move a calibration target device, the calibration target device comprising:

a plurality of cameras configured to capture an image of a target and acquire image information of the target, and

at least one measurer configured to acquire motion information of the calibration target device based on an acceleration of the calibration target device and a rotation direction of the calibration target device; and

at least one processor configured to:

match a first coordinate system, the first coordinate system corresponding to the plurality of cameras, to a third coordinate system,

match a second coordinate system, the second coordinate system corresponding to the at least one measurer, to the third coordinate system,

simultaneously perform, based on the third coordinate system, an image calibration operation with respect to the image information and a motion calibration operation with respect to the motion information,

generate calibration motion information via the image calibration operation with respect to the image information, and

generate calibration image information via the motion calibration operation with respect to the motion information,

wherein the third coordinate system is a standard coordinate system that unifies the first and second coordinate systems that are different from each other.

2. The calibration device of claim **1**, wherein the first coordinate system and the second coordinate system are matched to the third coordinate system by performing a coordinate transformation on the first coordinate system with respect to a position of the plurality of cameras in a first section and on the second coordinate system with respect to a position of the at least one measurer in the first section, the position of the plurality of cameras in the first section and the position of the at least one measurer in the first section being calculated through a simultaneous localization and mapping (SLAM) algorithm.

3. The calibration device of claim 1, wherein the at least one measurer is further configured to acquire the motion information in correspondence with a rotation or movement operation of the first calibrator.

4. The calibration device of claim 1, wherein the first calibrator comprises a mover capable of moving the first calibrator in at least one of an X-axis direction, a Y-axis direction, or a Z-axis direction.

5. The calibration device of claim 4, wherein the mover comprises a first mover capable of moving the first calibrator in the Z-axis direction and a second mover capable of moving the first calibrator in at least one of the X-axis direction or the Y-axis direction.

6. The calibration device of claim 1, further comprising:

a second calibrator comprising the target,

wherein the second calibrator is configured to move in at least one of an X-axis direction, a Y-axis direction, or a Z-axis direction.

7. The calibration device of claim 1, wherein the target comprises a repeated pattern image.

8. The calibration device of claim 1, further comprising a memory configured to store the calibration image information and the calibration motion information.

9. The calibration device of claim 1,

wherein the first calibrator is further configured to rotate or move a plurality of calibration target devices, and

wherein the calibration image information and the calibration motion information are simultaneously generated with respect to the plurality of calibration target devices.

10. A calibration method comprising:

capturing an image of a target;

acquiring image information of the target;

acquiring motion information of a calibration target device by measuring an acceleration of the calibration target device and determining a rotation direction of the calibration target device;

matching a first coordinate system, the first coordinate system corresponding to a plurality of cameras configured to capture the image of the target, to a third coordinate system;

matching a second coordinate system, the second coordinate system corresponding to a measurer configured to acquire the motion information of the calibration target device, to the third coordinate system;

simultaneously performing, based on the third coordinate system, an image calibration operation with respect the image information and a motion calibration operation with respect the motion information;

generating calibration motion information via the image calibration operation with respect to the image information; and

generating calibration image information via the motion calibration operation with respect to the motion information,

wherein the third coordinate system is a standard coordinate system that unifies the first and second coordinate systems that are different from each other.

11. The calibration method of claim 10, wherein the matching comprises performing a coordinate transformation on the first coordinate system with respect to a position of the plurality of cameras in a first section and on the second coordinate system with respect to a position of the measurer in the first section, the position of the plurality of cameras in the first section and the position of the measurer in the first section being calculated through a simultaneous localization and mapping (SLAM) algorithm.

12. The calibration method of claim 10, further comprising:

rotating or moving the calibration target device through a first calibrator,

wherein the acquiring of the motion information of the calibration target device comprises acquiring the motion information in correspondence with a rotation or movement operation of the first calibrator, and

wherein the acquiring is performed by the measurer.

13. The calibration method of claim 12, further comprising moving the first calibrator in at least one of an X-axis direction, a Y-axis direction, or a Z-axis direction.

14. The calibration method of claim 10, wherein the target moves in at least one of an X-axis direction, a Y-axis direction, or a Z-axis direction.

15. The calibration method of claim 10, wherein the target comprises a repeated pattern image.

16. The calibration method of claim 10, further comprising storing the calibration image information and the calibration motion information in a memory.

17. The calibration method of claim 10, wherein the generating of the calibration motion information and the calibration image information comprises simultaneously generating the calibration image information and the calibration motion information with respect to a plurality of calibration target devices.

18. A computer program product comprising a non-transitory computer readable recording medium comprising a program to perform operations of:

capturing a target;

acquiring image information of the target;

acquiring motion information of a calibration target device by measuring an acceleration of the calibration target device and determining a rotation direction of the calibration target device;

matching a first coordinate system, the first coordinate system corresponding to a plurality of cameras configured to capture an image of the target, to a third coordinate system;

matching a second coordinate system, the second coordinate system corresponding to a measurer configured to acquire the motion information of the calibration target device, to the third coordinate system;

simultaneously performing, based on the third coordinate system, an image calibration operation with respect to the image information and a motion calibration operation with respect to the motion information;

generating calibration motion information via the image calibration operation with respect to the image information; and

generating calibration image information via the motion calibration operation with respect to the motion information,

wherein the third coordinate system is a standard coordinate system that unifies the first and second coordinate systems that are different from each other.

19. The calibration device of claim 1, wherein the measurer comprises an inertial measurement unit (IMU).

* * * * *

Додаток Г. Параметри для завдань двовимірного позиціонування

Таблиця Е.1. Параметри для NEAT без пошуку новизни

[NEAT]	
fitness_criterion	max
fitness_threshold	10
pop_size	100
reset_on_extinction	False
no_fitness_termination	False
generations	1000
seed	1571021768
[Gym]	
environment	gym_robot_arm:robot-arm-v0
episode_reps	10
[DefaultGenome]	
num_hidden	1
num_inputs	4
num_outputs	7
activation_default	sigmoid
activation_mutate_rate	0.0
activation_options	sigmoid
aggregation_default	sum
aggregation_mutate_rate	0.0
aggregation_options	sum
bias_init_mean	0.0
bias_init_stdev	1.0
bias_max_value	30.0
bias_min_value	-30.0
bias_mutate_power	0.5
bias_mutate_rate	0.7
bias_replace_rate	0.1
compatibility_disjoint_coefficient	1.0
compatibility_weight_coefficient	0.5

conn_add_prob	0.5
conn_delete_prob	0.5
enabled_default	True
enabled_mutate_rate	0.01
feed_forward	True
initial_connection	full_nodirect
node_add_prob	0.2
node_delete_prob	0.2
response_init_mean	1.0
response_init_stdev	0.0
response_max_value	30.0
response_min_value	-30.0
response_mutate_power	0.0
response_mutate_rate	0.0
response_replace_rate	0.0
weight_init_mean	0.0
weight_init_stdev	1.0
weight_max_value	30
weight_min_value	-30
weight_mutate_power	0.5
weight_mutate_rate	0.8
weight_replace_rate	0.1
[DefaultSpeciesSet]	
compatibility_threshold	3.0
[DefaultStagnation]	
species_fitness_func	max
max_stagnation	20
species_elitism	1
[DefaultReproduction]	
elitism	2
survival_threshold	0.2
min_species_size	8
[Names]	

input	['position_x', 'position_y', 'joint_1', 'joint_2']
output	['0']

Таблиця Е.2. Параметри для NEAT з пошуком новизни

[NEAT]	
fitness_criterion	max
fitness_threshold	10
pop_size	100
reset_on_extinction	False
no_fitness_termination	False
generations	1000
seed	1571021768
[Gym]	
environment	gym_robot_arm:robot-arm-v0
episode_reps	10
[Novelty]	
k	10
threshold	0.3
limit	150
ndims	2
[DefaultGenome]	
num_hidden	1
num_inputs	4
num_outputs	7
activation_default	sigmoid
activation_mutate_rate	0.0
activation_options	sigmoid
aggregation_default	sum
aggregation_mutate_rate	0.0
aggregation_options	sum
bias_init_mean	0.0
bias_init_stdev	1.0

bias_max_value	30.0
bias_min_value	-30.0
bias_mutate_power	0.5
bias_mutate_rate	0.7
bias_replace_rate	0.1
compatibility_disjoint_coefficient	1.0
compatibility_weight_coefficient	0.5
conn_add_prob	0.5
conn_delete_prob	0.5
enabled_default	True
enabled_mutate_rate	0.01
feed_forward	True
initial_connection	full_nodirect
node_add_prob	0.2
node_delete_prob	0.2
response_init_mean	1.0
response_init_stdev	0.0
response_max_value	30.0
response_min_value	-30.0
response_mutate_power	0.0
response_mutate_rate	0.0
response_replace_rate	0.0
weight_init_mean	0.0
weight_init_stdev	1.0
weight_max_value	30
weight_min_value	-30
weight_mutate_power	0.5
weight_mutate_rate	0.8
weight_replace_rate	0.1
[DefaultSpeciesSet]	
compatibility_threshold	3.0
[DefaultStagnation]	
species_fitness_func	max

max_stagnation	100
species_elitism	1
[DefaultReproduction]	
elitism	2
survival_threshold	0.2
min_species_size	2
[Names]	
input	['position_x', 'position_y', 'joint_1', 'joint_2']
output	['0']

Додаток Д. Параметри для завдань тривимірного позиціонування з використанням зображень

[NEAT]	
fitness_criterion	max
fitness_threshold	10
pop_size	500
reset_on_extinction	False
no_fitness_termination	False
generations	1000
seed	1571021768
[Gym]	
environment	gym_robot_arm:robot-arm-v0
episode_reps	10
[Novelty]	
k	10
threshold	0.3
limit	150
ndims	2
[Substrate]	
input	[(0, 100), (0, 100), (0., 3.1), (0., 3.1)]
hidden	[[(-0.5, 0.5), (0.5, 0.5)], [(-0.5, -0.5), (0.5, -0.5)]]
output	[(0., 0.), (1., 1.), (2., 2.), (3., 3.), (4., 4.), (5., 5.), (6., 6.)]
function	sigmoid
[DefaultGenome]	
num_hidden	1
num_inputs	4
num_outputs	7
activation_default	sigmoid
activation_mutate_rate	0.0

activation_options	sigmoid
aggregation_default	sum
aggregation_mutate_rate	0.0
aggregation_options	sum
bias_init_mean	0.0
bias_init_stdev	1.0
bias_max_value	30.0
bias_min_value	-30.0
bias_mutate_power	0.5
bias_mutate_rate	0.7
bias_replace_rate	0.1
compatibility_disjoint_coefficient	1.0
compatibility_weight_coefficient	1.0
conn_add_prob	0.2
conn_delete_prob	0.02
enabled_default	True
enabled_mutate_rate	0.01
feed_forward	True
initial_connection	full_nodirect
node_add_prob	0.1
node_delete_prob	0.2
response_init_mean	1.0
response_init_stdev	0.0
response_max_value	30.0
response_min_value	-30.0
response_mutate_power	0.0
response_mutate_rate	0.0
response_replace_rate	0.0
weight_init_mean	0.0
weight_init_stdev	1.0
weight_max_value	30
weight_min_value	-30
weight_mutate_power	0.5

weight_mutate_rate	0.5
weight_replace_rate	0.1
[DefaultSpeciesSet]	
compatibility_threshold	1.9
[DefaultStagnation]	
species_fitness_func	max
max_stagnation	100
species_elitism	1
[DefaultReproduction]	
elitism	1
survival_threshold	0.3
min_species_size	5
[Names]	
input	['position_x', 'position_y', 'joint_1', 'joint_2']
output	['0']

Додаток Е. Список публікацій здобувача за темою дисертації

1. Вітюк, А. Є., Корнага, Я. І., & Барабаш, А. О. (2018). Захоплення невідомих об’єктів мобільним роботом із використанням візуальної інформації. Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки, (29 (68),№ 1 (1)), 93-98 (Фахове видання, «Б»).
2. Вітюк, А. Є., & Дорошенко, А. Ю. (2022). Програмний пакет для оцінки похибки калібрування стереокамери в системі комп’ютерного зору. Проблеми програмування (3-4), 469-477 (Фахове видання, «Б»).
3. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Пошук новизни у методі нейроеволюції для позиціонування роботизованої кінцівки. Проблеми програмування (3), 49-57 (Фахове видання, «Б»).
4. Вітюк, А. Є., & Дорошенко, А. Ю. (2023). Програмний пакет для адаптивного навчання контролерів роборуки на основі нейромереж. Проблеми програмування (4), 98-107 (Фахове видання, «Б»).
5. Корнага Я.І., Вітюк А.Є. (2018). Оцінка якості стійкого захвату, спланованого на основі тривимірної реконструкції цільового об’єкту по зображенням з монокамери, Міжнародна науково-технічна конференція «The International Conference on Security, Fault Tolerance, Intelligence».
6. Alona Vitiuk, Anatoliy Doroshenko. (2022) Software Package for Evaluation the Stereo Camera Calibration for 3D Reconstruction in Robotics Grasping System. CEUR Workshop Proceedings Proceedings of the 13th International Scientific and Practical Programming Conference UkrPROG
7. Вітюк А.Є., Дорошенко А.Ю. (2023). Використання нейроеволюції при пошуку політик в формі нейромереж для управління робочою кінцівкою. Перша Науково-теоретична конференція «Пріоритети і виклики реалізації Стратегії розвитку штучного інтелекту в Україні». Artificial Intelligence (2).
8. Вітюк А.Є., Корнага Я.І. (2018). Оцінка впливу похибок калібрування на тривимірну реконструкцію у монокулярній системі одночасної локалізації та картографування. Східноєвропейський науковий журнал (4 (32)), 29–35.

9. US10841570B2, Petro Kytsun Iegor Vdovychenko, Alona Vitiuk, Nataliya Sakhnenko, Oleksii Panfilov, «Simultaneous camera and sensors calibration apparatus and method», 2018.