УДК 62-5-681.5

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ РУХУ РОБОТА-ПИЛОСОСА З КОРЕКЦІЄЮ ШЛЯХУ

Л. В. Мельнікова[1], К.В. Коваль[1], С. В. Ніченко[2]
[1] Державний університет «Одеська політехніка»
[2]Інститут Пауля Шеррера

Анотація. *На базі мови програмування Python розроблено середовище тестування системи керування роботом-пилососом, заснованої на нечіткій логіці з поступовим вдосконаленням коду програми та його алгоритму. Результати моделювання дозволяють оцінити ефективність від впровадження кожного вдосконалення – від найпростішого використання нечіткої логіки, що дозволяє лише уникати перешкоди на шляху робота, до алгоритму з використанням внутрішньої пам'яті з метою покращення стабільності роботи.*

Ключові слова: *середовище моделювання PYTHON, нечітка логіка, кінематика робота-пилососа, дефазифікація, метод центру ваги*

Вступ

Сучасні технології з кожним днем захоплюють все більші області діяльності людини. І це вже давно стало стосуватися не тільки виробничої складової автоматизації, а й прийшло в наш побут. Інноваційні системи створюють новий, якісний рівень комфорту нашого повсякденного життя. Хвиля автоматизації не оминула й такий примітивний прилад як пилосос.

Автоматизація побутових приладів забезпечує зручність і скорочує час, що витрачається на домашні справи. Відмінною рисою робота-пилососа є його автономність. Провідні виробники дивують нас новинками у вдосконаленні, такими як розпізнавання не тільки власного розташування робота-пилососа, але і конкретних навколишніх предметів, в тому числі і рухомих об'єктів.

Основною проблемою роботів-пилососів є задача навігації. При вирішенні проблеми планування шляху є два основних моменти: запобігання зіткненню з перешкодами і зменшення часу для досягнення мети або економія енергії. Для вирішення цих проблем багато дослідників розробили різні методи, включаючи алгоритм пошуку A*, штучні потенційні поля, нечітку логіку, нейронні мережі, генетичні алгоритми, алгоритм оптимізації наслідуванням мурашиної колонії і т. д., в тому числі і комбінації цих методів. Наприклад, гібрид генетичного алгоритму і нечіткої логіки [7], що дає кращий результат з точки зору часу в шляху і середньої швидкості.

У даній статті ми надаємо порівняльне дослідження трьох алгоритмів, заснованих на методі нечіткої логіки центру ваги, а також розробку середовища моделювання в Python для безпосереднього їх тестування і отримання наочних результатів.

Метою роботи є проектування середовища тестування для робота-пилососа, впровадження різних моделей його поведінки в довільних середовищах та порівняння результатів з метою оцінки ефективності використаних алгоритмів.

1. Вирішення проблеми

Надійність навігації в динамічному або невідомому середовищі залежить від здатності роботів переміщатися між невідомими перешкодами без зіткнень і швидкої реакції на невизначеності. Нечітка логіка використовує евристичні знання для подання та реалізації методології для розробки заснованих на сприйнятті дій для навігації мобільних роботів. Крім того, методологія нечіткої логіки дуже корисна при роботі з невизначеностями в реальному світі, і точна модель середовища не є абсолютно необхідною для навігації. Тому на основі простої конструкції, простих властивостей реалізації та надійності нечіткої логіки було розроблено багато підходів для вирішення проблеми навігації мобільного робота в відстеженні цілей, відстеженні шляху, обході перешкод, координації поведінки та моделюванні середовища. [3]

Щоб підвищити загальну продуктивність навігаційної системи, складні навігаційні завдання розбиваються на ряд більш простих і більш дрібних підсистем (поведінок), які називаються системами на основі поведінки. У заснованій на поведінці системі кожна поведінка отримує певну сенсорну інформацію і перетворює її в визначену відповідь. Поведінка може включати в себе відстеження шля-

ху, обхід перешкод, відстеження цілі, досягнення мети і т.д. Нарешті, на основі вихідних даних команди активної поведінки робот виконує дію. [2]

Проблеми, пов'язані з поведінковими навігаційними системами, – це координація поведінки або вибір дій. Множинна поведінка може одночасно генерувати кілька командних виходів, що може привести до руху робота в непередбачених напрямках або до повної відмови системи. Надійна робота системи залежить від рішення про те, як інтегрувати високорівневе планування і низькорівневу поведінку при виконанні, яку поведінку слід активувати і як вихідні команди повинні бути об'єднані в одну команду для управління роботом.

2. Матеріали і результати досліджень

2.1. Кінематика колес робота-пилососа

Розглянемо кінематику колес робота-пилососа та рівняння, які описують його рух. Робот з диференціальним приводом має два мотори, по одному на кожне колесо (на рисунку – це великі колеса). Зміна напрямку руху досягається за рахунок різних швидкостей (звідси і назва – диференційний) [6]. Рух колес робота зображено на рис. 1.

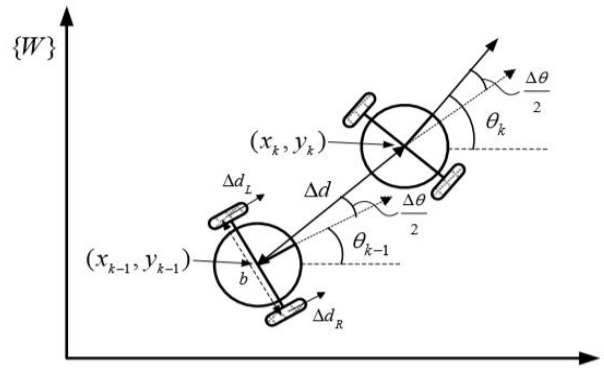

Рис. 1. Кінематична модель робота пилососа

Для прямолінійного руху колеса повинні обертатися з однаковими швидкостями. Для того, щоб робот розвернувся на місці, необхідно встановити швидкості однаковими по модулю, але спрямованими протилежно. Інші комбінації швидкостей призводять до руху по дузі. Позиціонування одометрії робота диференціального привода може бути обчислена наступними рівнянням:

$$\begin{aligned} x_k &= x_{k-1} + \Delta d \cdot \cos(\theta_{k-1} + \Delta\theta / 2) \\ y_k &= y_{k-1} + \Delta d \cdot \sin(\theta_{k-1} + \Delta\theta / 2) \\ \theta_k &= \theta_{k-1} + \Delta\theta \end{aligned} \qquad (1)$$

де*:*

x_k: Положення робота в напрямку *x* в момент часу *k*;

y_k: Положення робота в напрямку *y* в момент часу *k*;

θ_k: Провідний напрямок робота в момент часу *k*;

Інкрементальне зміщення правого та лівого коліс Δd:

$$\Delta d = \frac{\Delta d_R + \Delta d_L}{2} \qquad (2)$$

Інкрементальний кут напрямку робота $\Delta\theta$:

$$\Delta\theta = \frac{\Delta d_R - \Delta d_L}{b} \qquad (3)$$

Номінальний діаметр правого (лівого) колеса $d_{R(L)}$:

$$\Delta d_{R(L)} = \pi D_{R(L)} \frac{N_{R(L)}}{\mathrm{Re}_{R(L)}}, \qquad (4)$$

де:

$D_{R(L)}$- Номінальний діаметр правого (лівого) колеса;

$N_{R(L)}$- Кількість імпульсів датчика правого (лівого) колеса;

$Re_{R(L)}$- Дозвіл датчика правого (лівого) колеса.

Звідси:

$$\Delta d_R = \Delta d - \frac{b\Delta\theta}{2}$$

$$\Delta d_L = \Delta d + \frac{b\Delta\theta}{2}.$$

Ці рівняння знайшли своє використання в коді програми.

2.2. Модулі програми мови Python

У коді програми були використані стандартні бібліотеки Array та NumPy.

-Модуль Array визначає масиви в Python.

- NumPy – це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць та операцій з цими масивами.

- PIL – Python Imaging Library, це бібліотека мови Python, що може бути використана для роботи з зображеннями.

- Matplotlib – це бібліотека мови Python, що використовується для візуалізації даних двовимірної (2D) та тривимірної (3D) графіки. [5]

- Код програми містить різноманітні модулі, кожен з яких виконує відведену участь. Такими модулями є:

- Controls – основи логіки, базовий клас;

- Fuzzy – універсальний модуль роботи з нечіткою логікою;

- Room – модуль оцифровування рисунків приміщення

- Animation – відповідає за візуальну складову – рух робота, побудова карти, побудова графіків;
- System – загальний клас середовища моделювання, відповідає за взаємодію всіх об'єктів в системі; відповідає за симуляцію;
- Utils – допоміжні утиліти та функції;
- Sensor – відповідає за роботу умовних сенсорів програмованого робота-пилососа. Містить датчик дистанції та датчик кута повороту;
- Robot – відповідає за зчитування даних з умовних датчиків, аналіз та реакцію на ці дані; містить функцію пам'яті, за допомогою якої вибудовується карта приміщення у самій пам'яті робота;
- Simulation – основний модуль, який відповідає за симуляцію. З цього модулю запускається код програми, у тому числі:
    - запуск оцифровування рисунку приміщення (Room);
    - блок, що реалізує використання нечіткої логіки;
    - покроковий розрахунок шляху робота;
    - запуск модулю Animation.

2.3. Метод нечіткої логіки центру ваги

Найбільші переваги, які надає використання нечіткої логіки понад іншими системами, – це можливість оперувати вхідними даними, заданими нечітко: наприклад, що безупинно змінюються в часі значення (динамічні задачі), значення, що неможливо задати однозначно, а також можливість проведення швидкого моделювання складних динамічних систем і їхній порівняльний аналіз із заданим ступенем точності [8].

Вибір системи нечіткої логіки обґрунтований тим, що вона ідеально підходить для вирішення завдання навігації, як безпосередньо робота-пилососа, так і будь-якого іншого мобільного робота, що вимагає навігації в просторі: він простий у використанні і реалізації в програмному коді. За своєю суттю нагадує використання функції якщо... то (IF…Then), але з набагато більшими можливостями по розподілу та використанню вхідних величин. Це в свою чергу дає можливість плавного регулювання вихідними величинами, в нашому випадку – кутом повороту, а також швидкістю пересування.

Метод центру ваги (CENTER OF GRAVITY (COG)) забезпечує чітке значення на основі центру ваги нечіткого набору. Загальна площа розподілу функції приналежності, що використовується для подання комбінованої дії управління, ділиться на ряд підобластей. Площа та центр ваги кожної підобласті обчислюється, а потім приймається підсумовування всіх цих піддіапазонів для знаходження дефазифікованого значення для дискретної нечіткої множини [4].

Метод центру ваги представлений у вигляді модуля Fuzzy.Py та у самій симуляції (Simulation.py) в класі логіки (class Logic (Logic Base)) у функції Setup. Використаний модуль Fuzzy.Py, є універсальним. Його можна використовувати в будь-якому програмному коді, так як уявляє собою запрограмовані правила нечіткої логіки, що використовують Метод центру ваги.

2.4. Етапи моделювання

1 Перший етап. Найпростіша модель поведінки. На першому етапі роботи був створений універсальний модуль роботи з нечіткою логікою (Fuzzy.py). Це найпростіша модель, так як використовує для генерації руху робота лише завданий алгоритм: вибудовує правила нечіткої логіки та проявляє реакцію на стіни та об'єкти, що з'являються в області дії датчиків. При наближенні до стіни або об'єкту робот змінює свою швидкість та кут повороту.

2 Другий етап. Введення станів робота Normal та Stuck. На другому етапі роботи до найпростішої моделі поведінки робота додається два стани: Normal та Stuck. При нормальному стані робота рух здійснюється як і на першому етапі. Стан Stuck означає застрявання робота. Наприклад, у вузькому просторі або у куті, коли роботу не вистачає заданого куту повороту, щоб вибратися з закутку самостійно. Цей стан виявляється за допомогою датчиків за умовами, вказаними в програмному коді в модулі Robot.Py.

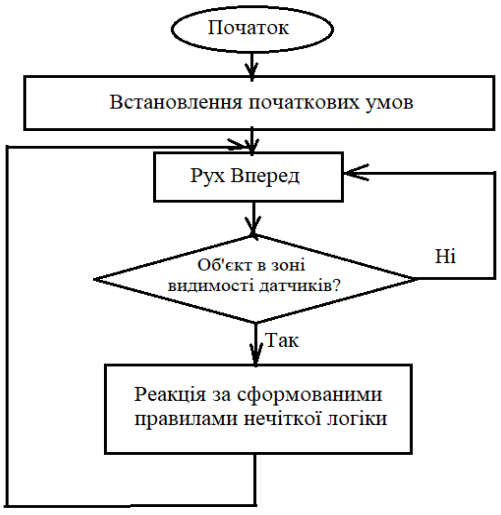

Рис. 2. Алгоритм роботи програми для першого етапу

3 Третій етап. Введення корекції шляху. На третьому етапі роботи до перших двох етапів додається функція корекції. В модулі Simulation ця функція представлена однією строчкою. Суть корекції розписана в класі Memory модуля Robot.Py і представлена функцією getDirection. Ця функція дозволяє скорегувати алгоритм поведінки робота. За допомогою класу пам'яті (Memory) він тепер може не тільки визначати свій стан, а й реагувати на пройдений шлях і відрізняти його від непройдених ділянок мапи. Таким чином вдосконалюється алгоритм побудови шляху робота, надається перевага ділянкам, де він ще не був.

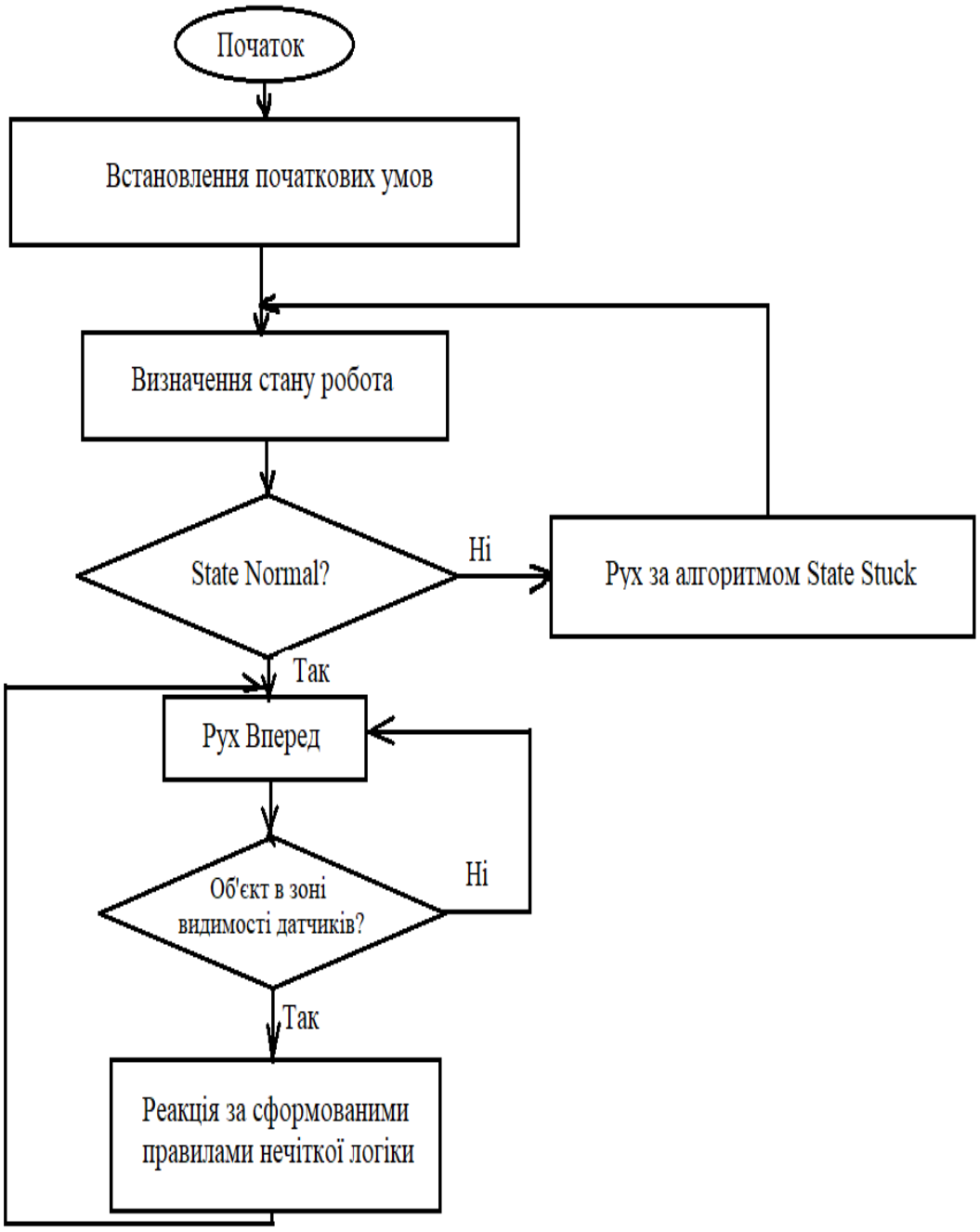

Рис. 3. Алгоритм роботи програми для другого етапу

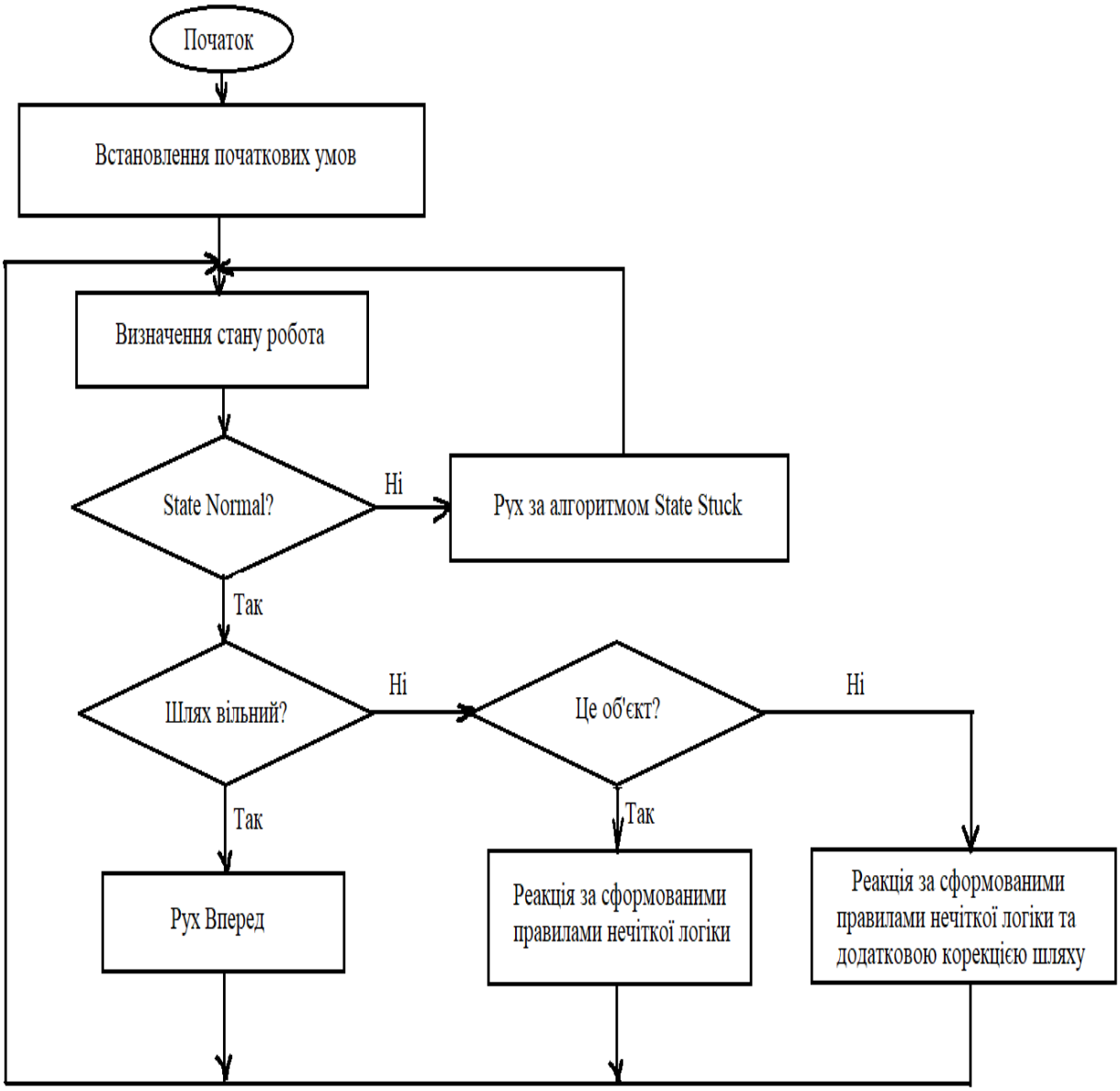

Рис. 4. Алгоритм роботи програми для третього етапу

2.5. Результати тестування

Збір результатів почнемо з того, що побудуємо кілька гіпотез:

- перша гіпотеза полягає в тому, що в порожній кімнаті перший і другий алгоритм буде працювати приблизно однаково, так як в порожній кімнаті не буде існувати умов для застрявання. Таким чином, крім поведінки в кутах, алгоритм руху другого алгоритму буде схожий з першим.

- друга гіпотеза полягає в тому, що в другій кімнаті перший алгоритм буде видавати помилки, тому що в цій кімнаті є багато місць, де робот може застрягти (наприклад, в кутку кімнати справа внизу).

- третя гіпотеза полягає в тому, що в третій кімнаті перший алгоритм буде вести себе некоректно. Спеціально були залишені незафарбовані ділянки, щоб побачити, чи не заскочить туди робот, незважаючи на стіни, які повинні обмежити його рух.

- четверта гіпотеза полягає в тому, що при алгоритмі третього етапу роботи отримані результати будуть вищими, ніж у перших двох, незалежно від кімнати.

Ці гіпотези визначають наші очікування від виконання програми. Для коректної оцінки результатів зберемо статистичний аналіз по всіх трьох кімнатах кожного алгоритму. Для цього прогонимо алгоритми в кожній кімнаті по 10 разів. Для наочності по кожному алгоритму в кожній кімнаті наведемо по 3 результати: найгірший, середній і кращий.

В якості результатів представимо 2 рисунки: візуальне відображення кімнати, робота і його шляху в цій кімнаті, а також графік відсотка покриття площі приміщення. Всі кімнати, взяті для тестування робота, мають розміри 150 х 150 пікселів. Умовний час тестування робота становить 30 хвилин. Розміри кімнати в перекладі на реальні розміри складають 5 х 5м^2.

Кімната №1 – найпростіша. Представляє собою квадрат 5 х 5 м^2 без об'єктів всередині.

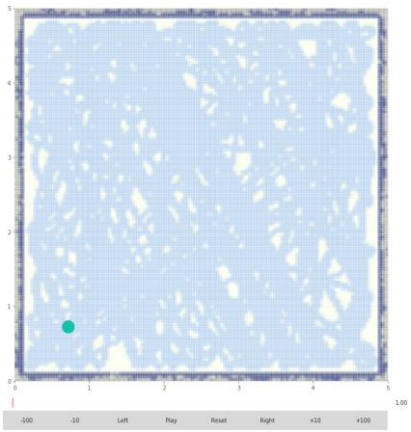

Рис. 5. Найкращий результат першого алгоритму. Покрита поверхня становить 84%

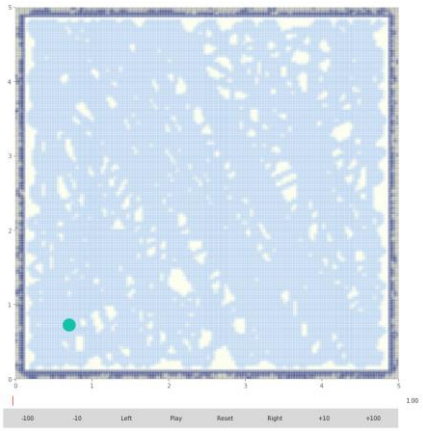

Рис. 6. Найкращий результат другого алгоритму. Покрита поверхня становить 86%

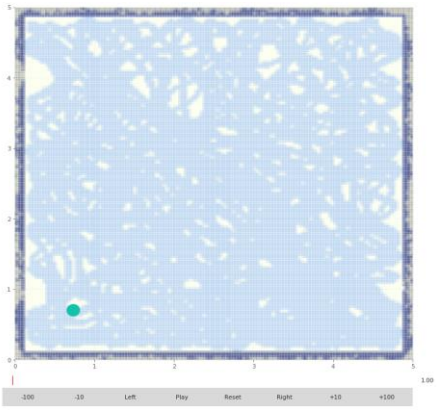

Рис. 7. Найкращий результат третього алгоритму. Покрита поверхня становить 86%

Кімната №2 – складна. Представляє собою квадрат 5 х 5 м^2 з об'єктами всередині та великою кількістю перешкод.

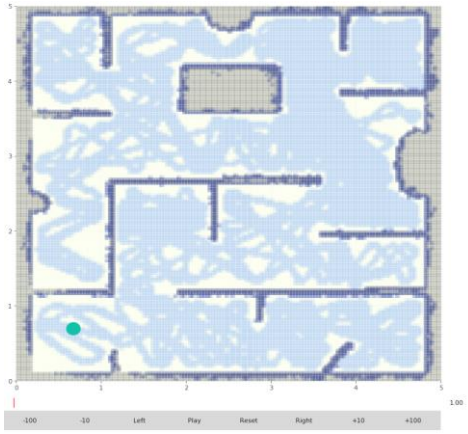

Рис. 8. Найкращий результат першого алгоритму. Покрита поверхня становить 66%

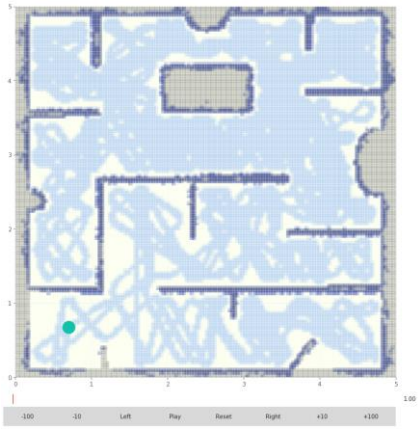

Рис. 9. Найкращий результат другого алгоритму. Покрита поверхня становить 64,5%

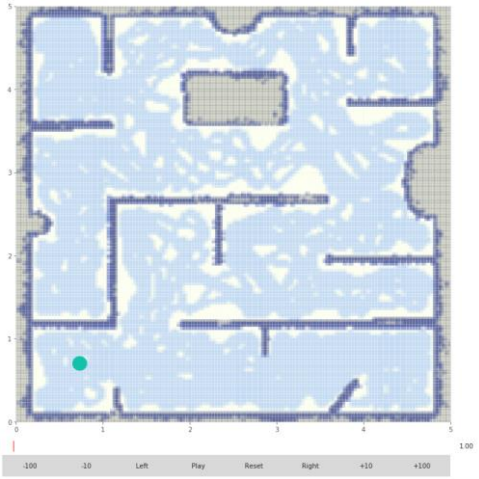

Рис. 10. Найкращий результат третього алгоритму. Покрита поверхня становить 73%

Кімната №3 – складна. Представляє собою квадрат 5 х 5 м^2 зі стінами довільної форми та незафарбованими ділянками.

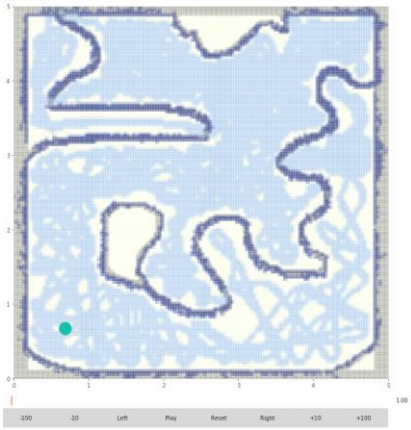

Рис. 11. Найкращий результат першого алгоритму. Покрита поверхня становить 64%

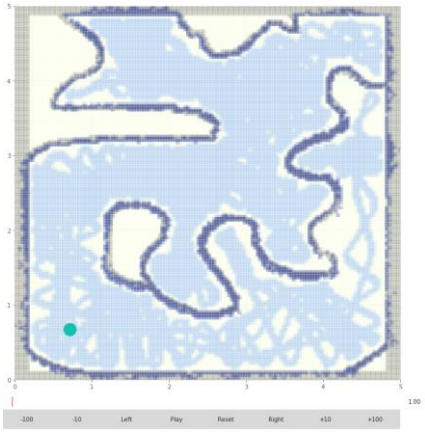

Рис. 12 - Найкращий результат другого алгоритму. Покрита поверхня становить 61%

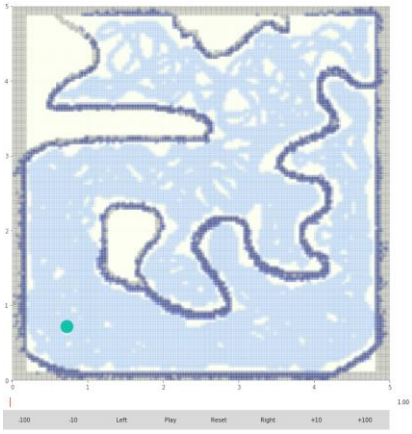

Рис. 13. Найкращий результат третього алгоритму. Покрита поверхня становить 65%

3. Аналіз отриманих результатів

Для проведення статистичного аналізу результатів було проведено 90 модуляцій програмного коду, а саме: по кожному з трьох алгоритмів (простий алгоритм з нечіткою логікою, нечітка логіка з уникненням застрівання, нечітка логіка з реакцією робота на власний пройдений шлях) в кожній з трьох кімнат було проведено по 10 модуляцій. Зібрані результати представлені у таблицях нижче.

Перша кімната: середнє арифметичне області покриття по першому алгоритму – 82,6%, по другому алгоритму – 83,4%, по третьому алгоритму – 84,6%.

Таблиця 1

Статистика по першій кімнаті

№ алгоритму / № тесту	1	2	3
1	84%	81%	83%
2	80%	84%	85%
3	82%	85%	86%
4	84%	82%	85%
5	83%	85%	84%
6	84%	83%	84%
7	79%	84%	86%
8	84%	80%	84%
9	83%	84%	85%
10	83%	86%	84%

Таблиця 2

Статистика по другій кімнаті

№ алгоритму / № тесту	1	2	3
1	27%	58%	49,5%
2	65%	64,5%	56%
3	57%	55%	71%
4	35%	67%	58%
5	66%	56%	68%
6	46%	53%	63%
7	63%	57%	68%
8	56%	56%	72%
9	66%	59%	70%
10	43%	32%	73%

Таблиця 3

Статистика по третій кімнаті

№алгоритму / №тесту	1	2	3
1	27%	58%	49,5%
2	65%	64,5%	56%
3	57%	55%	71%
4	35%	57%	68%
5	66%	56%	68%
6	46%	53%	63%
7	63%	57%	68%
8	56%	56%	72%
9	66%	59%	70%
10	43%	32%	73%

Друга кімната: середнє арифметичне області покриття по першому алгоритму – 52,4%, по другому алгоритму – 65,85%, по третьому алгоритму – 65,85%.

Третя кімната: середнє арифметичне області покриття по першому алгоритму – 40,825%, по другому алгоритму – 54%, по третьому алгоритму – 60,9%

Як і очікувалося, в першій кімнаті майже немає різниці між використаними алгоритмами шляху. Проте статистика показує, що все ж таки вибір алгоритму впливає на область покриття, хоч і не в значній мірі. Статистичний аналіз другої кімнати показав, що перший алгоритм працює нестабільно, на відміну від другого та третього. За статистикою по області покриття здається, що другий та третій алгоритми майже не відрізняються один від одного. Проте слід зауважити, що в цій кімнаті було багато збоїв моделювання. Тобто насправді було проведено набагато більше операцій, ніж 10, особливо для першого алгоритму. Майже кожен другий запуск першого алгоритму давав збій, що свідчить про те, що робот застрягав в закутках кімнати та намагався вийти за її межі. Другий алгоритм виявляв збій модуляції три рази, в той час як третій алгоритм не видавав жодної помилки.

В третій кімнаті, як було зазначено наперед одні в третій гіпотезі, перший алгоритм вів себе некоректно, заскакуючи в пусте місце на мапі кімнати. Таким чином, хоча процент покриття і так малий (40,825%), насправді він становить ще меншу частину, так як програма розраховує статистику і по цим не зафарбованим ділянкам. Другий алгоритм дозволив роботу не заскакувати за встановлені межі, проте з задачею покриття він впорався лише на 54%, в той час як третій алгоритм впорався з задачею на 60,9%. Враховуючи незафарбовані ділянки, в цій кімнаті можливо досягти площі покриття лише 74%.

Порівнюючи всі три алгоритми виявляємо наступне: перший алгоритм дозволяє досягти непоганих результатів у простому просторі, проте зовсім не пристосований до роботи в складних кімнатах с розмаїттям закутків за меблів. Другий алгоритм працює краще, проте, як виявилось в другій кімнаті, також може збоїти і застрягати, хоч і набагато менше, ніж при застосуванні першого алгоритму. Третій алгоритм на всіх ділянках, в тій чи іншій мірі, показує кращі результати, хоча б на кілька процентів. За всі 10 моделювань не було жодного збою. А також, якщо подивитись на статистику проценту покриття кімнати, можна зауважити, що в порівнянні з іншими алгоритмами, алгоритм роботи на нечіткій логіці с корекцією напрямку в залежності від пройденого шляху працює набагато стабільніше.

Висновки

У цьому дослідженні використана система нечіткої логіки з мінімальним набором правил, необхідних для якісної навігації в просторі, що дозволяє також покривати найбільшу поверхню. Проблема обходу перешкод при навігації робота-пилососа вирішена шляхом розробки трьох алгоритмів (які поступово удосконалюються) системи виведення нечіткої логіки.

Розроблене середовище моделювання дозволило провести симуляцію поведінки робота-пилососа за всіма трьома алгоритмами. В управлінні роботом в моделюванні досягається досить задовільна продуктивність.

Результати моделювання повністю підтвердили доцільність та ефективність запропонованих методів по вирішенню проблеми навігації робота-пилососа в завідомо невідомому просторі та уникання перешкод.

Треба зазначити, що запропоновані перші два алгоритми також мають місце для практичного використання. Хоча вони й менш ефективні, проте, як показує моделювання в простіших віртуальних середовищах, можуть бути використані у відповідних умовах, тобто в приміщеннях менш обставлених об'єктами.

За рахунок скорочення часу, що витрачається пилососом на чистку поверхні, також знижується споживання електроенергії.

Список використаної літератури

1. Kooktae Lee, Changbae Jung and Woojin Chung Accurate calibration of kinematic parameters for two wheel differential mobile robots. [Electronic Resource],– 2011.
2. Tobias Edwards, Jacob Sörme. A Comparison of Path Planning Algorithms for Robotic Vacuum Cleaners, [Electronic Resource], –2018.
3.Kazi Mahmud Hasan, Abdullah-Al-Nahid, Khondker Jahid Reza. Path planning algorithm development for autonomous vacuum cleaner robots [Electronic Resource], –2014.
4. Dragan Z. Saletic, Dusan M. Velasevic, Nikos E. Mastorakis. Analysis of Basic Defuzzification Techniques, . [Text], Belgrade, Yugoslavia and Piraeus, –2002.
5.https://www.ibm.com/developerworks/ru/library/l-python_part_5/index.html [Electronic Resource]
6. http://robotosha.ru/robotics/robot-motion.htm l. [Electronic Resource].
7. Algabri M., Mathkour H., Ramdane H. and Alsulaiman M., Comparative Study of Soft Computing Techniques for Mobile Robot Navigation in an Unknown Environment. . [Text] Computers in Human Behavior, –2015, 50: 42–56.
8. https://studopedia.su/8_59033_zagalna-struktura-nechitkogo-mikrokontrolera.html [Electronic Resource].

References

1. Kooktae Lee, Changbae Jung and Woojin Chung (2011) Accurate calibration of kinematic parameters for two wheel differential mobile robots.
2. Tobias Edwards, Jacob Sörme (2018) A Comparison of Path Planning Algorithms for Robotic Vacuum Cleaners.
3.Kazi Mahmud Hasan, Abdullah-Al-Nahid, Khondker Jahid Reza (2014) Path planning algorithm development for autonomous vacuum cleaner robots.
4. Dragan Z. Saletic, Dusan M. Velasevic, Nikos E. Mastorakis (2002) Analysis of Basic Defuzzification Techniques, Belgrade, Yugoslavia and Piraeus.
5.https://www.ibm.com/developerworks/ru/library/l-python_part_5/index.html.
6. http://robotosha.ru/robotics/robot-motion.html.
7. Algabri M., Mathkour H., Ramdane H. and Alsulaiman M., (2015), Comparative Study of Soft Computing Techniques for Mobile Robot Navigation in an Unknown Environment. Computers in Human Behavior, 50: 42–56.
8. https://studopedia.su/8_59033_zagalna-struktura-nechitkogo-mikrokontrolera.html.

ALGORITHMIZATION AND MOTION PROGRAMMING OF THE VACUUM CLEANER-ROBOT WITH PATH CORRECTION

L. V. Melnikova[1], K. V. Koval[1], S. V. Nichenko[2]
[1] Odessa Polytechnic State University
[2]Paul Scherrer Institute

Abstract. *The reliability of navigation in a dynamic or unknown environment depends on the ability of robot vacuum cleaners to move between unknown obstacles without collisions and rapid response to uncertainty. Problems associated with behavioral navigation systems are behavior coordination or choice of actions. Multiple behavior can simultaneously generate multiple command outputs, which can lead to the movement of the robot in unforeseen directions or to the complete failure of the system. Reliable system performance depends on deciding how to integrate high-level scheduling and low-level execution behavior, which behavior should be activated, and how the source commands should be combined into a single robot management command.*

The choice of fuzzy logic system is justified by the fact that it is ideal for navigation, as for a robot vacuum cleaner itself, and any other mobile robot that requires navigation in space: it is easy to use and implement in software code. In essence, it resembles the use of the function IF… Then, but with much greater opportunities for the distribution and use of input values. This in turn allows for smooth adjustment of the output values, in our case – the angle of rotation, as well as speed.

This research uses the CENTER OF GRAVITY (COG) method, which provides a clear value based on the center of gravity of a fuzzy set and is considered the most accurate of all defasification methods.

This article presents the results of the development of an environment for simulating the movement of a robot vacuum cleaner, which allows one to estimate the differences between the applied algorithms for using fuzzy logic and the reliability of the chosen method. It is also possible to evaluate the effectiveness of

the embedded logic for a path-choosing. In the simulation, it is possible to import a room layout, which gives an idea of the need to implement complex logic.

***Key words**: PYTHON modeling environment, fuzzy logic, robot-vacuum cleaner kinematics, defasification, center of gravity method, simulation.*

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ ДВИЖЕНИЯ РОБОТА-ПЫЛЕСОСА С КОРРЕКЦИЕЙ ПУТИ

Л. В. Мельникова[1], Е. В. Коваль[1], С. В. Ниченко[2]
[1] Государственный университет «Одесская политехника».
[2]Інститут Пауля Шеррера

***Аннотация.** На базе языка программирования Python разработана среда тестирования системы управления роботом-пылесосом, основанная на нечеткой логике с постепенным совершенствованием кода программы и его алгоритма. Результаты моделирования позволяют оценить эффективность от внедрения каждого совершенствования - от простого использования нечеткой логики, которое позволяет лишь избегать препятствия на пути робота, к алгоритму с использованием внутренней памяти с целью улучшения стабильности работы.*

***Ключевые слова:** среда моделирования PYTHON, нечеткая логика, кинематика робота-пылесоса, дефаззификация, метод центра тяжести.*

Отримано 17.04.2021

Мельнікова Любов Василівна, Державний університет "Одеська політехніка", кандидат технічних наук, доцент кафедри електромеханічної інженерії. Просп. Шевченка,1, Одеса, 65044, Україна, E-mail: lubov.v.melnikova@gmail.com , т.+38-099-677-6374

Melnikova Lubov, Odessa Polytechnic State University, Doctor of Philosophy (Ph. D), Associate Professor of Department of Electromechanical Engineering,. Shevchenko ave., 1, Odessa, 65044, Ukraine. E-mail: lubov.v.melnikova@gmail.com, т.+38-099-677-6374

ORCID ID: 0000-0002-1732-1930

Коваль Катерина Василівна, Державний університет "Одеська політехніка", студентка 1 курсу магістратури кафедри електромеханічної інженерії. Просп. Шевченка,1, Одеса, 65044, Україна,
E-mail: kat.vas.koval@gmail.com , т.+38-099-147-4830

Koval Kateryna, Odessa Polytechnic State University, 1st year student of the Master's Degree at the Department of Electromechanical Engineering,. Shevchenko ave., 1, Odessa, 65044, Ukraine.
E-mail: kat.vas.koval@gmail.com , т.+38-099-147-4830
ORCID ID: 0000-0001-5489-0326

Ніченко Сергій Володимирович, департамент ядерної енергетики та безпеки інституту Паула Шеррера, кандидат тахнічних наук, науковий співробітнік. Forschungsstrasse 111, 5232 Філліген PSI, Швейцарія
E-mail: sergii.nichenko@psi.ch тел. +41 76 474 02 26

Sergii Nichenko, Nuclear Energy and Safety (NES) Department at the Paul Scherrer Institut, Doctor of Philosophy (Ph.D.), research scientist. Forschungsstrasse 111, 5232 Villigen PSI, Switzerland
E-mail: sergii.nichenko@psi.ch тел. +41 76 474 02 26
ORCID ID: 0000-0001-7700-2141