СИСТЕМНИЙ АНАЛІЗ

УДК 004.42, 004.02, 004.94 *http://dx.doi.org/10.30970/vam.2024.34.00000*

МОДЕЛЮВАННЯ АВТОНОМНОГО КЕРУВАННЯ РУХОМ РОБОТА НА ПЛОЩИНІ ЗА УМОВИ НЕВИЗНАЧЕНОГО РОЗМІЩЕННЯ ПЕРЕШКОД

В. Черняхівський

Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79000, Україна
e-mail: volodymyr.chernyakhivskyy@lnu.edu.ua

Досліджуємо питання автономного керування рухом робота (формального виконавця) на площині з перешкодами. Площину прямокутної форми моделюємо матрицею квадратних ділянок (кліток), на окремих ділянках розташовані перешкоди. Задаються стартова та фінішна позиції (клітка) робота. Розміщення перешкод для робота апріорі невідоме, в поточній позиції він може визначити перешкоди тільки на суміжних ділянках. Робот може рухатись в кожному з восьми напрямків, в позиції з перешкодами рух неможливий. У момент підходу до перешкоди робот аналізує стан суміжних кліток та визначає напрям руху шляхом відповідного повороту. Мета виконавця – переміститися зі стартової позиції в задану фінішну позицію. Вважаємо, що розміщення перешкод дозволяє побудувати потрібний шлях. За умови опуклості перешкод (в термінах кліток) та їх ізольованості побудовано евристичний алгоритм автономного керування рухом робота, який забезпечує його переміщення в задану фінішну позицію. Для мінімізації кількості кроків (пройдених кліток) запропоновано "правило мінімального квадрата". Визначено команди аналізу стану робота і середовища, команди руху, аналітичні команди. Всі команди подано у вигляді методів відповідних класів для задач автономного керування рухом робота. Запропонований алгоритм реалізовано у вигляді програмного комплексу для моделювання та візуалізації руху робота.

Ключові слова: робот, навігація, площина, модель, шлях, команда, алгоритм.

1. Вступ

Переміщення автономно керованого робота на площині (чи в просторі) є актуальною темою в теперішніх обставинах сучасного світу. Автори публікацій розглядають питання дослідження і оцінки загальної схеми пошуку шляху переміщення. Описані такі аспекти як математична модель, умови отримання розв'язку задачі, точність розв'язку, ефективність методу, особливості застосування, порівняльна характеристика.

Так, в роботі [1] розглядають здатність автономного транспортного засобу дістатися до цільової локації, подорожуючи через невідоме середовище. Це питання досліджується шляхом програмування мобільного робота для пошуку найкоротшого маршруту у лабіринті. Розглядають декілька типових конфігурацій лабіринту і алгоритми пересування, які побудовані на основі хвильового алгоритму. Подібні дослідження є в роботі [2], де представлений модифікований алгоритм, який припускає, що невідомі стіни можуть бути проникними. В роботі [3] розглядають автономну навігацію робота в системі лабіринту на основі модифікованого стеження за стіною, щоб розв'язати проблему нескінченного повернення назад.

Розглядають також питання особливих умов руху робота [4]. Досліджується оптимальне планування шляху робота в невідомому середовищі при відсутності або наявності небезпечного простору. Небезпека визначається як ділянка, яка не є перешкодою, але потенційно ризикована. В основі статті розглядають технічні аспекти планування руху на основі сигналів датчиків і оцінки ситуації.

Планування шляхів антропоморфних роботів в невідомих середовищах розглянуто в роботі [5]. В статті пропонують вирішити проблему відповідно до нечітких марковських процесів прийняття рішень з урахуванням простору. Робот може пересуватись горизонтально, вертикально і діагонально. Інформацію роботу надає камера спостереження за навколишнім середовищем, яка дає $3D$-матрицю даних, які можуть змішуватись з шумом. В роботі розглядають аналіз рішень на основі політик залежних від функції дії. В подібному напрямку є робота [6], в якій представлено розробку та впровадження систем нейронного керування мобільними роботами для обходу перешкод у реальному часі з використанням ультразвукових датчиків. Запропоновано кілька способів технічного вирішення проблеми на основі нейронних мереж і контролера навігації, і відповідну конструкцію робота.

В статті [7] розглянуто в першу чергу технічні аспекти будови безпілотних літальних апаратів (БЛА) і проведено аналіз параметрів, які впливають на вибір команд керування БЛА. Проблему керування БЛА запропоновано вирішувати на основі застосування апарату нечіткої логіки для керування траєкторією руху. Контролери апаратури генерують сигнали, які можна вважати аналогами команд керування рухом. БЛА пересувається в тривимірному просторі і має команду задання висоти польоту, яка стосується вертикального переміщення. Якщо виконати горизонтальний переріз тривимірного простору площиною, тоді решта команд керування стосуються керуванням у двомірному просторі, що є еквівалентом площини. В роботі показані також результати керування горизонтальним переміщенням на основі задання кута нахилу.

Стаття [8] провадить аналіз систем будови маршрутів БЛА у залежності від різних параметрів, таких, як мінімізація витрат енергії, зниження рівня шуму, мінімізація часу польоту. Відповідно подано огляд індивідуальних алгоритмів вирішення проблем. Розглядають моделі і алгоритми, як мають такі параметри, як змінна околиця, заборонений рух, обмеження довжини маршруту, розподіл задач між кількома БЛА, пошук шляху, планування маршруту, гібридні перевезення дроном та вантажівкою, наземне автопілотування, датчики виявлення перешкод. Моделі і алгоритми розділено на дві групи. До першої групи належать ті, які будують маршрут на стороні інформаційної системи. Друга група забезпечує динамічну будову маршруту на основі технічних засобів безпосередньо БЛА. В статті проведено огляд елементів алгоритмів будови маршруту.

В статті [9] розглянуто хвильовий алгоритм з модифікаціями для визначення траєкторій польоту БЛА. Розроблена методика визначення траєкторії на основі показників видимості чи невидимості, ймовірності виявлення цілі, розмірів хмари ефективного спостереження, маски місцевості як матриці кодування ділянок з коефіцієнтами прохідності. Хвильовий алгоритм корегується до тривимірного простору при моделюванні поширення хвилі. Маску місцевості запропоновано розширити до тривимірної за рахунок третього індексу дискретизації простору по висоті. В роботі виконано аналіз будови модифікованого хвильового алгоритму.

В підсумку на рисунку 1 показано основний зміст підходів до розв'язування задач пошуку шляху формальних виконавців на площині. На рис. 1, *а* шлях пересування

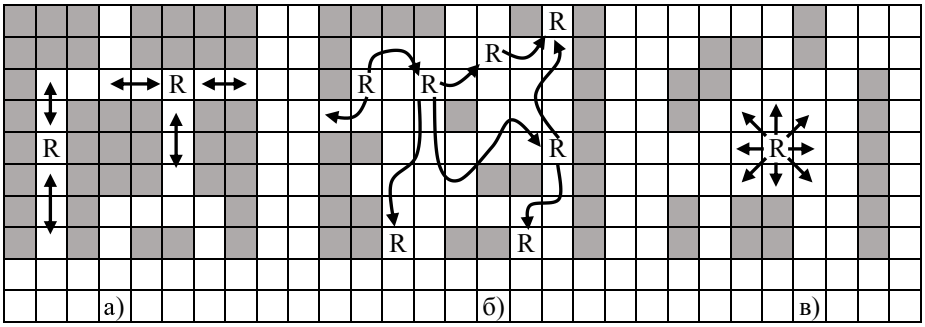

Рис. 1. Принципи організації руху робота

визначений конфігурацією лабіринту, пересування в кожній позиції лабіринту можливі лише вздовж коридору, тобто в двох напрямках, або поворотами до іншого коридору. Задачі розв'язують методами аналізу лабіринтів. Робот в кожній позиції має інформацію про ближнє оточення – стіни, коридор, а також зібрану інформацію про раніше відвідані позиції.

На рис.1,*б* пересування відбувається від деякої фіксованої позиції до іншої фіксованої позиції на основі відомої структури вузлів, доріг (сполучень), даних технічних пристроїв, і моделюється графами і алгоритмами на графах. Таке пересування в загальному випадку потребує попередньої інформації про вузли і дороги, і, можливо, попередньої оцінки.

На рис.1,*в* показано ідею іншого підходу до пересування формального виконавця (робота, дрона). Будуємо конструктивну систему керування формальним виконавцем для руху площиною за дотриманням таких визначених умов. Площину моделюємо не переходами, які дозволені, а навпаки – фіксуємо частини площини, де пересування неможливе – перешкоди. При цьому небезпечні ділянки так само вважаємо перешкодами в базовому варіанті дослідження. Формальний виконавець може рухатись в кожному з восьми напрямків, якщо є вільна позиція. В кожній позиції відома конфігурація перешкод лише довкола виконавця, отже невідомий загальний план площини. Задача виконавця така сама – досягнути потрібної позиції в результаті пошуку шляху за мінімальну кількість кроків. Виконавцю достатньо вміти виконувати один крок пересування, керувати напрямом руху і мати простий датчик ідентифікації перешкоди в ближньому оточенні.

У поданій статті запропонована система команд керування роботом для виконання поставленого перед ним завдання, побудовані і досліджені алгоритми прийняття рішень і поведінки робота в процесі руху на основі правила мінімального квадрата, лівостороннього і правостороннього обходу, оптимізації вибору напряму. Система команд орієнтована на оцінку ближнього до робота стану середовища і на швидке прийняття рішення. Площина пересування обмежена для досліджень, але з практичної точки зору можна вважати площину достатньо великою в лінійному вимірі. Крім того, моделюється пересування з врахуванням географічних напрямів, це дозволяє за потреби зв'язати шлях з топографічною картою місцевості.

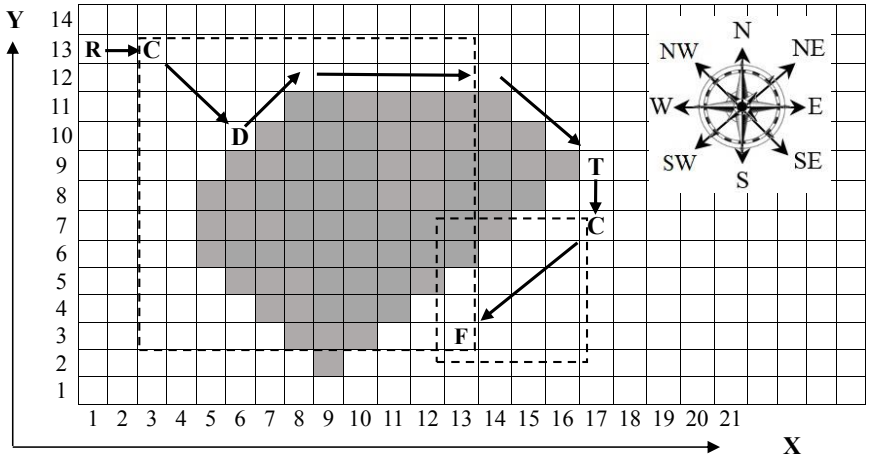

Рис. 2. Елементи дискретної моделі площини і руху робота

2. Постановка задачі

Для складання алгоритмів пересування площиною треба мати дискретну модель площини. Площину прямокутної форми моделюємо матрицею квадратних ділянок, як показано на рис.2. Зразу зауважимо, що ступінь дискретності не впливає на загальні міркування, викладені в статті.

Позначення на рис.2 і кодування для числового моделювання:
порожня клітка – позиція площини вільна від перешкод і об'єктів (код 0);
зафарбована клітка – на позиції площини розташована перешкода (код 1);
R початкова позиція виконавця (код 2);
F позиція фінішу – мета пересування виконавця (код 3);
C позиція зміни напряму руху на частині ділянки без перешкод (код 4);
D позиція підходу до перешкоди, місце початку процедури обходу (код 5);
T позиція закінчення процедури обходу перешкоди (код 6), дальший рух відбувається за траєкторією без перешкод.

Штриховою лінією показані розрахункові мінімальні квадрати, які будемо використовувати для оптимізації шляху пересування і потенційного збільшення швидкості руху.

Кожний квадрат площини може бути вільним, або належати якомусь об'єкту, або бути використаним в обчисленні траєкторії руху. Якщо квадрат вільний – можна надати йому значення 0. Якщо квадрат належить якомусь об'єкту, як показано на рис.2, тоді надаємо йому значення відповідно до типу об'єкта. Наприклад, позиції перешкод можуть мати значення 1, стартова – значення 2, а фінішна – 3. Інші кодування визначаємо за практичною реалізацією обчислювальної процедури траєкторії руху.

Будемо вважати, що площина належить першій чверті декартової площини координат. Тоді координати кожного квадрата є звичними: $[1,1]$, $[1,2]$, $[1,3]$, $[2,1]$, $[3,1]$, $[2,2]$, і так далі. Зауважимо, що нумерація починається від 1, а не від 0, бо мова йде не про відстань від початку координат, а про позицію квадратів.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 0 0 1 1 1 0 0 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Рис. 3. Файл вхідних даних

Для моделювання алгоритмів пересування вважаємо, що вихід за межі заданої площини не допускається.

Відповідно до рисунку 2 файлове зображення за вхідними даними, тобто початкова задана конфігурація, в текстовому форматі буде виглядати так, як на рисунку 3.

Такі файли можна будувати на основі, наприклад, топографічних карт і отриманої зовнішньої інформації. Зауважимо, що тут мова йде про дослідження правил руху робота, на практиці робот не знає позиції і конфігурацію перешкод.

Формулювання задачі. Нехай дана прямокутна область площини, на якій в різних позиціях є певні перешкоди, неможливі для пересування. Розташування перешкод має довільні позиції. Будова кожної перешкоди задовільняє таким вимогам: 1) не сполучена з будь-якою іншою – завжди є прохід між сусідніми перешкодами; 2) має замкнений власний периметр і є суцільною; 3) є опуклою за відомим математичним визначенням; якщо сполучити прямою лінією дві сусідні клітинки оболонки, тоді ціла перешкода має бути по одну сторону від лінії; або: плоска фігура опукла, якщо їй належить відрізок, що сполучає будь-які дві точки; властивість опуклості є важливою для будови алгоритмів пересування, викладених в статті; 4) розміри і конфігурація перешкод довільні, але границі перешкод не дотикаються до границь ділянки, завжди існує прохід між межею ділянки і перешкодою; 5) розміри і межі цілої ділянки є не фізичними, а логічними, які ми фіксуємо як граничні умови розв'язку задачі.

Зауважимо, що при виконанні перерахованих вимог до будови перешкод завжди існує шлях від стартової позиції до фінішної.

Побудувати систему команд робота, визначити правила пересування, скласти алгоритм переходу площиною від заданої початкової позиції до фінішної позиції. Алгоритм має забезпечити обхід ділянок з перешкодами і мінімізацію кількості пройдених клітинок. Ро́боту наперед не відоме розташування перешкод, робот може ідентифікувати перешкоду, лише перебуваючи в сусідній клітці, тобто, підійшовши впритул до перешкоди. Проте робот завжди знає координати свого поточного розташування і координати мети. Робот може ідентифікувати об'єкти лише на

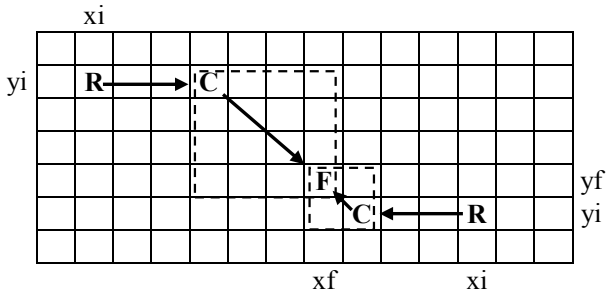

Рис. 4. Елементи пересування на ділянці без перешкод

близькій відстані довкола свого поточного розташування – горизонтально, вертикально чи діагонально.

Проєктування системи команд робота виконуємо з метою моделювання методів застосування команд і для дослідження алгоритмів пересування. В результаті отримуємо можливість оцінювання команд і алгоритмів за різними критеріями.

3. Алгоритм будови маршруту пересування

Запропонований алгоритм є евристичним, він побудований на апріорно визначених правилах руху. Як кожний евристичний алгоритм, він не гарантує мінімального шляху, але завжди дозволяє його знайти, використовуючи правила оптимізації.

Визначимо допустимі напрямки руху горизонтально, вертикально і діагонально:

- North / N вверх (північ);
- NorthEast / NE направо вверх (північний схід);
- East / E направо (схід);
- SouthEast / SE направо вниз (південний схід);
- South / S вниз (південь);
- SouthWest / SW вниз наліво (південний захід);
- West / W наліво (захід);
- NorthWest / NW наліво вверх (північний захід).

Приймемо за основу пересування за критерієм мінімізації довжини шляху, за яким треба з поточної позиції завжди переходити на сусідню в напрямку вектора позиції мети, якщо така сусідня позиція вільна.

На рисунку 4 показано елементи пересування на ділянці без перешкод для двох випадків початку руху: East і West. (x_i, y_i) – поточні координати робота, (x_f, y_f) – координати позиції мети.

Для оптимізації довжини шляху використаємо "правило мінімального квадрата". Мінімальним назвемо квадрат за стороною $\min(|x_f - x_i|, |y_f - y_i|)$, одна з вершин якого належить позиції мети F, а діагонально протилежна має бути на тій самій горизонталі чи вертикалі (залежно від конфігурації), що й поточна позиція робота. На рисунку 4 штриховою лінією показані розрахункові мінімальні квадрати. Від стартової чи поточної позиції R рух виконуємо горизонтально або вертикально до позиції вершини мінімального квадрата. Далі рух продовжується діагонально до позиції фінішу F. Такий поділ на максимально довгі прямолінійні ділянки руху дозволяє зменшити кількість маневрів за рахунок поворотів. Напрями і ділянки

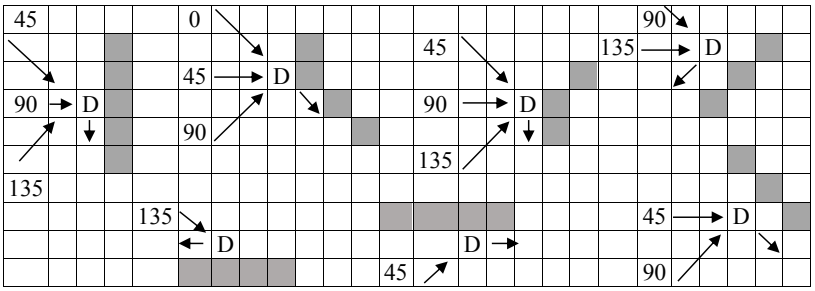

Рис. 5. Деякі варіанти початку правостороннього обходу

руху показані стрілками. Оскільки модель площини апроксимована квадратами, а переходити можна лише на сусідній квадрат, мінімальний шлях на кожному окремому кроці завжди буде горизонтально, вертикально чи діагонально в сторону позиції фінішу або вздовж контуру перешкоди.

На кожному кроці потрібно пам'ятати напрямок руху.

Пропонуємо такий алгоритм автономного керування рухом робота:

1. Якщо є вільна позиція в напрямку вектора позиції фінішу (x_m, y_m), то перейти до неї; напрямок щоразу обчислювати за схемою, показаною вище; якщо досягнули позиції фінішу – кінець алгоритму.

2. Якщо позиція попереду напряму руху належить перешкоді, тоді перейти до процедури правостороннього обходу перешкоди. Використаємо для цього відоме "правило лівої руки" яке застосовують наприклад, в задачах пересування лабіринтом. Для цього, підійшовши до перешкоди, повертаємо направо за годинниковою стрілкою на 45, 90 або 135 градусів відповідно до контуру перешкоди, тримаючись умовно лівою рукою за перешкоду.

На рисунку 5 показані для прикладу деякі конфігурації для випадків, коли робот підходить до краю перешкоди і перебуває в позиції D, рухаючись напрямком E, SE, NE. Стрілками показані напрями, якими потрапили до краю перешкоди і яким напрямком треба продовжити рух для випадку методу правостороннього обходу. Напрями наступного кроку руху можуть мати варіанти, залежно від точної конфігурації границі перешкоди і від наших правил руху. Числами позначені необхідні повороти в градусах за годинниковою стрілкою. Нуль означає, що можна продовжити рух в тому самому напрямку.

Якщо робот підходить до краю перешкоди від верхнього боку площини (напрямки S, SE, SW), достатньо повернути рисунок 5 на 90 градусів за годинниковою стрілкою. При цьому відповідно будуть повернуті конфігурації країв перешкоди і стрілки напрямів.

Аналогічні міркування можна продовжити для випадків підходу робота з правого боку площини і від нижнього боку площини.

Щоб визначити наближення робота до перешкоди, треба виконати перевірку конфігурації сусідніх граничних кліток перешкоди.

Тримаючись умовно лівою рукою краю перешкоди (при наближенні до перешкоди впритул і після повороту направо перешкода буде ліворуч), продовжуємо рух кроками по одній клітці вздовж границі перешкоди, повторюючи її контур і виконуючи відповідні повороти. Цю частину руху моделюємо окремим алгоритмом.

3. Рух вздовж стіни перешкоди продовжуємо доти, поки рух не відновиться в напрямку позиції C мінімального квадрата, як показано на рис.2 (позиція T); позицію мінімального квадрата треба обчислювати щоразу після кожного кроку.

4. Повернутись до кроку 1 і продовжити рух; забезпечуємо обхід наступної перешкоди, якщо така буде.

4. Система команд виконавця

Прийнявши до уваги викладені вище правила руху на площині з перешкодами, треба формалізувати систему команд виконавця, за якою можна реалізувати правила руху. Система команд – це окремі елементи алгоритму виконавця (робота). Систему команд можна будувати різними способами. Далі описано будову системи команд, яка передбачає логічні групи команд. Важливим є лиш те, щоб система команд була функціонально повною, тобто, надавала можливість будови шляху будь-якої конфігурації.

Для випадку виконавця-робота систему команд можна розділити на три частини: 1) команди аналізу стану виконавця і середовища; 2) команди руху; 3) аналітичні команди. Покажемо основні команди.

Команди аналізу стану виконавця і середовища:

GetStartFinish() – знайти на карті початкові координати і координати фінішу робота;

SetCurrentDir(yfrom, xfrom, yto, xto) – визначити напрям руху за координатами двох точок – від робота до мінімального квадрата чи іншої точки;

getnormvisBearings(y,x,bearings) – отримати нормалізовану ділянку карти розміром 3x3 за напрямком *bearings*, перебуваючи в позиції y, x;

getnormvisAdjacently(y,x,bearings) – отримати нормалізовану ділянку карти розміром 3x3 за напрямком *bearings*, в околі позиції y, x ліворуч і праворуч;

decision(self,y,x,bearings) – розпізнавання перешкоди, обчислити тип перешкоди за моделлю;

detour(bearings,lhrh,y,x) – обхід перешкоди в околі позиції y, x, обчислити тип дії за моделлю; *bearings* – напрям руху; *lhrh* – вид обходу "LH" чи "RH".

Команди руху:

Starting() – початкові налаштування руху;

GetTheRobotsWay() – запуск процедури пошуку шляху;

Step1() – перейти на сусідню клітку в напрямку руху, напрям зберігається;

TestObstacleAhead() – перевірка на перешкоду попереду і поворот.

Аналітичні команди: (комплексний аналіз)

LeftHandBypass() – лівосторонній обхід;

RightHandBypass() – правосторонній обхід.

Зазначені команди програмно реалізовано як методи класів для задач будови шляхів пересування робота. Програмування задач пересування площиною з перешкодами спирається на такі методи класів (функції). Самі функції за необхідності можна вдосконалити, наприклад, створити інший алгоритм для аналітичних команд правостороннього чи лівостороннього обходу. При цьому буде збережений основний алгоритм загального обчислення шляху.

Насамкінець зазначимо, що описані правила руху і сам алгоритм пересування є евристичними. Можливість експериментів з правилами руху і алгоритмом пересування залишається доступною до вдосконалення в програмній системі підтримки

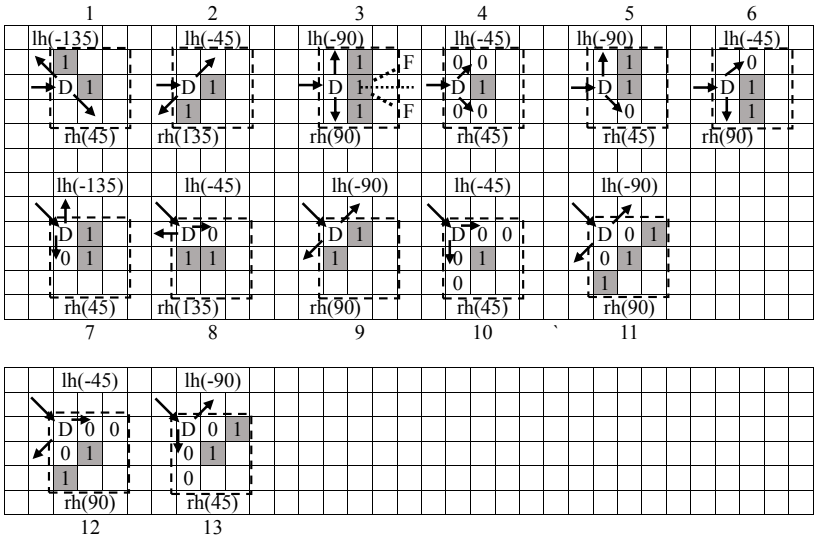

Рис. 6. Визначення напряму обходу і кута повороту

запропонованого підходу вирішення задачі.

5. Прийняття рішень в процесі руху

5.1. Моделі розпізнавання перешкод

При розпізнаванні лише встановлюємо новий напрям руху шляхом повороту, пересування тут не виконуємо, а будемо виконувати пересування за моделями обходу. Повороти ліворуч lh позначаємо від'ємними кутами в градусах, а повороти праворуч rh – додатними кутами в градусах.

Визначення напряму обходу і кута повороту в момент підходу до перешкоди показано на рис.6 для випадків напрямків руху E і SE, інші випадки отримуємо шляхом обертання на 90, 180 чи 270 градусів.

D – поточна позиція робота. Стрілка, яка входить до позиції D, визначає напрям, звідки був зроблений крок. Стрілки, які виходять від позиції D, показують можливі напрямки дальшого руху.

Штриховою лінією позначена матриця 3x3 даних позиції, отримана командою (функцією) *GetTheStatusInFront()*. Ця команда доступна для робота в кожній позиції його розташування. rh – правосторонній обхід, lh – лівосторонній обхід, число – кут повороту (додатний направо, від'ємний наліво). Приймаємо до уваги будову перешкод за правилом опуклості, як однієї з умов моделювання площини. Значення кліток для прийняття рішення: 1 – перешкода, 0 – вільно, без підпису – не впливає на рішення (чи значення випливає з правила опуклості).

Зауважимо, що показані напрями обходу визначені евристично і можуть будувати не найкоротший шлях, але завжди його знаходять. Для отримання найкоротшого цілого шляху треба знати наперед розташування і конфігурацію всіх перешкод на площині, а за початковою постановкою задачі така інформація недоступна роботу під час пересування.

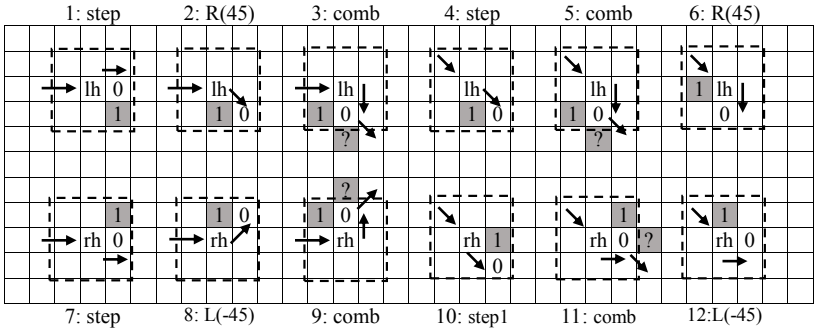

Рис. 7. Обчислення напрямку наступного кроку

5.2. Моделі обходу перешкод

Для обходу перешкоди виконуємо аналіз стану сусідніх клітоk за функцією *GetTheStatusAdjacently()*. За кожним тактом робимо або один крок step вперед (обидві стрілки в одному напрямку), або поворот R чи L (стрілки в різних напрямках), або комбінацію comb декількох дій. За підсумком такту поворот не можна виконати на кут, більший ніж 45 градусів, щоб не пропустити момент збігання поточного напрямку з напрямком в сторону мінімального квадрата – це ознака закінчення обходу перешкоди.

Спосіб обчислення напрямку наступного кроку показано на рисунку 7 для випадків лівостороннього lh і правостороннього rh обходу.

lh – лівосторонній обхід і поточна позиція робота, rh – правосторонній обхід і поточна позиція робота, R (кут) чи L (кут) – поворот направо чи наліво на кут в градусах, comb – комбінація декількох кроків.

Комбінації comb декількох кроків мають такий склад:
для випадку 3 виконати підряд 3 дії: R (90), step1, L (-45), бо невідома клітка за межами;
для випадку 5 виконати підряд 3 дії: R (45), step1, L (-45);
для випадку 9: L (-90), step1, R (45);
для випадку 11: L (-45), step1, R (45).

6. Лівосторонній обхід

Зберігаємо основний алгоритм будови маршруту пересування. Міняємо лише спосіб обходу перешкоди – замість правостороннього використаємо лівосторонній.

В будові алгоритму пересування міняємо пункт 2 – замість правостороннього обходу визначаємо лівосторонній. Використаємо для цього "правило правої руки" яке застосовують в задачах пересування лабіринтом. Для цього, підійшовши до перешкоди, повертаємо наліво за годинниковою стрілкою на 45, 90 або 135 градусів (рис. 8).

На рис. 8 показані для прикладу деякі конфігурації для випадків, коли робот підходить до краю перешкоди за напрямками E, SE, NE. Стрілками показані напрями, якими потрапили до краю перешкоди і яким напрямком треба продовжити

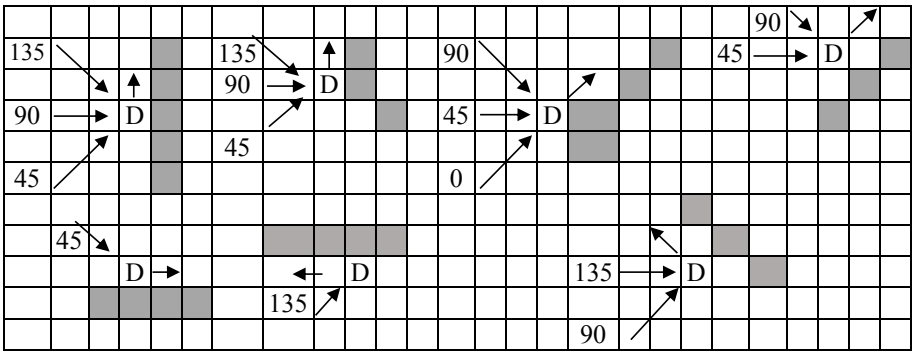

Рис. 8. Деякі варіанти початку лівостороннього обходу

рух. Числами позначені необхідні повороти в градусах проти годинникової стрілки. Тримаючись умовно правою рукою стіни перешкоди (при наближенні до перешкоди впритул і після повороту наліво перешкода буде праворуч), продовжуємо рух кроками по одній клітці вздовж границі перешкоди, повторюючи її контур і виконуючи відповідні повороти.

7. Приклад розв'язку задачі

При виконанні кожної команди переміщення робота генерується інформація про параметри команди в форматі ((XYpos), (position status), (bearings,command)). (XYpos) визначає поточну позицію робота, (position status) визначає характеристику (код) позиції, (bearings,command) визначають азимут руху і виконану команду.

Розроблена програмна система моделювання руху робота може бути застосована в двох варіантах. За першим варіантом робот отримує лише координати своєї початкової позиції і координати позиції мети. Робот пересувається автономним керуванням шляхом оцінки поточної ситуації і генерування необхідних команд, як описано в статті.

За другим варіантом рух можна обчислити наперед як послідовність команд переміщення робота на основі відомої карти місцевості і перешкод, виконати незалежно і зовні, так само використовуючи наявні команди робота. В цьому разі робот отримує завдання на переміщення (карту навігації) і просто його реалізує. В процесі переміщення робот зіставляє команди навігації з реальним станом середовища пересування. Якщо виявлено невідповідність, тоді робот переходить в режим автономного керування і продовжує рух до місця досягнення мети за алгоритмом аналізу перешкод.

Приклад знайденого шляху руху показано на рисунку 9. Шлях позначений послідовністю чорних квадратів і літер ситуацій, які трапляються на шляху.

За таким шляхом отримаємо протокол руху, початковий фрагмент:
$(0, 0, -1, begin)$, $(4, 4, 2, start)$, $(20, 17, 3, finish)$, $(4, 4, 2, E - setdir)$, $(5, 4, 9, E - step1)$, $(6, 4, 9, E - step1)$, $(6, 4, 5, NW - turn)$, $(6, 4, 5, NW - D)$, $(5, 5, 9, NW - step1)$, $(4, 6, 9, NW - step1)$, $(4, 6, 9, N - turn)$, $(4, 6, 9, NE - turn)$, $(5, 7, 9, NE - step1)$, $(5, 7, 9, E - turn)$, $(5, 7, 6, E - end - D)$, $(5, 7, 6, E - T)$, $(6, 7, 9, E - step1)$, $(7, 7, 9, E - step1)$, $(8, 7, 9, E - step1)$, $(8, 7, 5, NE - turn)$, $(8, 7, 5, NE - D)$, $(9, 8, 9, NE - step1)$, ...

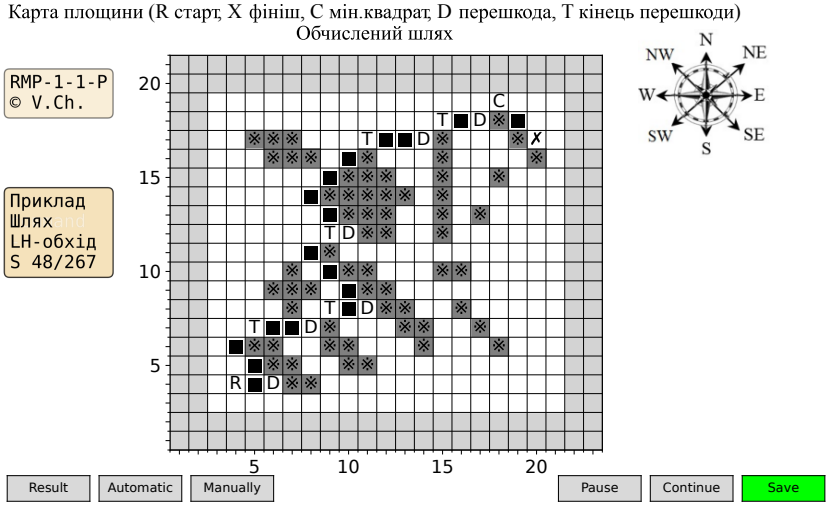

Рис. 9. Приклад знайденого шляху руху

8. Висновки

У межах прийнятих припущень побудовано алгоритм та відповідний програмний комплекс RMP-1-1-P (robot movement along a plane, prospector, version 1-1) для моделювання автономного керування рухом робота. У запропонованому варіанті реалізовано пошук шляху за правилом "мінімальних квадратів" і руху в напрямку фінішної позиції, алгоритмів лівостороннього і правостороннього обходу, а також евристичної оцінки напрямку обходу перешкод. В програмній реалізації використано технології дискретизації площини, руху за командами, правила мінімального квадрата, розпізнавання дискретних об'єктів, зв'язування з географічною картою, програмування на python, візуалізація результатів з використанням matplotlib. Алгоритм та його програмна реалізація протестовані на конкретному прикладі по моделюванню процесу автономного керування рухом робота.

В навчальних цілях основні ідеї статті застосовано для підготовки завдань для студентів магістерського рівня, як окрема тема курсу "Алгоритмічні моделі інформатики" для спеціальності "Середня освіта (Інформатика)" [10].

Список використаної літератури

1. Saman Abu Bakar S. Solving a Reconfigurable Maze using Hybrid Wall Follower Algorithm / Abu Bakar Sayuti Saman, Issa Abdramane // International Journal of Computer Applications (0975–8887). – November 2013. – Vol. 82, № 3. – , P. 22–26. – Available: https://research.ijcaonline.org/volume82/number3/pxc3892114.pdf.
2. CaiZ. FloodFill Maze Solving with Expected Toll of Penetrating Unknown Walls for Micromouse / Z. Cai, L. Ye, A. Yang // In 2012 IEEE 14th International Conference on High Performance Computing and Communications. – 2012. – P. 1428–1433. – Available: https://ieeexplore.ieee.org/document/6332345.
3. AlamriS. An Autonomous Maze-Solving Robotic System Based on an Enhanced Wall-Follower Approach / S. Alamri, H. Alamri, W. Alshehri, S. Alshehri, A. Alaklabi, T. Alh-

miedat, // Machines. – 2023. – Vol. 11. – 249. – Available: https://www.mdpi.com/2075-1702/11/2/249.

4. Jahanshahi Hadi Robot Motion Planning in an Unknown Environment with Danger Space / Hadi Jahanshahi, Mohsen Jafarzadeh, Naeimeh Najafizadeh Sari, Viet-Thanh Pham, Van Van Huynh, Xuan Quynh Nguyen // Electronics. – 2019. – Vol. 8 (2), 201. – Available: https://www.mdpi.com/2079-9292/8/2/201.
5. Fakoor Mahdi Humanoid robot path planning with fuzzy Markov decision processes / Mahdi Fakoor, Amirreza Kosari, Mohsen Jafarzadeh // Journal of Applied Research and Technology. – 2016. – Vol. 14. – P. 300–310. Available: https://jart.icat.unam.mx/index.php/jart/article/view/20/20.
6. Medina-Santiago A. Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors / A. Medina-Santiago, J.L. Camas-Anzueto, J.A. Vazquez-Feijoo, H.R. Hernandez-de Leon, R. Mota-Grajales // Journal of Applied Research and Technology. – February 2014, – Vol. 12, Iss. 1. – P. 104–110. – Available: https://www.sciencedirect.com/science/article/pii/S1665642314716104.
7. Лисенко С.М. Інтелектуалізована система керування безпілотними літальними апаратами / С.М. Лисенко, С.В. Румянцев // Міжнародний журнал Комп'ютерні системи та інформаційні технології. – 2020. – № 1. – С. 21–27. – Available: https://csitjournal.khmnu.edu.ua/index.php/csit/article/view/7/3.
8. Кучеренко О.І. Огляд досліджень щодо системи побудови маршрутів дронів / О.І. Кучеренко, Т.А. Вакалюк // Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. – Інформатика, обчислювальна техніка та автоматизація. – 2024. – Т. 35 (74), № 1. – С. 178–184. – Available: https://www.tech.vernadskyjournals.in.ua/journals/2024/1_2024/part_1/29.pdf.
9. Даник Ю.Г. Хвильовий метод визначення траєкторій БПЛА для ефективного спостереження за ділянкою місцевості / Ю.Г. Даник, І.І. Балицький // Сучасні інформаційні технології у сфері безпеки та оборони. – Т. 33, № 3 (2018). – С. 143–147. – Available: https://sit.nuou.org.ua/article/view/154401/154367.
10. Черняхівський В.В. Силабус навчальної дисципліни «Алгоритмічні моделі інформатики» / Черняхівський В.В. // Львівський національний університет імені Івана Франка. – Факультет прикладної математики та інформатики. – Кафедра програмування. – Львів, 2024. – С.1–9. – Available: https://ami.lnu.edu.ua/wp-content/uploads/2020/11/Alh_modeli_Sylabus_SO-24-3.pdf.

Стаття: надійшла до редколегії 21.08.2025
доопрацьована 24.09.2025
прийнята до друку 08.10.2025

MODELING OF AUTONOMOUS CONTROL OF WORKING ON A PLANE UNDER CONDITIONS OF UNDEFINED OBSTACLE LOCATION

V. Chernyakhivskyy

Ivan Franko National University of Lviv,
1, Universytetska str., 79000, Lviv, Ukraine,
e-mail: volodymyr.chernyakhivskyy@lnu.edu.ua

We investigate the issue of autonomous control of the movement of a robot (formal executor) on a plane with obstacles. We model a rectangular plane with a matrix of square sections (cells), obstacles are located in separate sections. The starting and finishing positions (cells) of the robot are given. The placement of obstacles is unknown a priori for the robot, in the current position it can determine obstacles only in adjacent sections.

The robot can move in each of eight directions, in a position with obstacles movement is impossible. At the moment of approaching an obstacle, the robot analyzes the state of adjacent cells and determines the direction of movement by making a corresponding turn. The goal of the executor is to move from the starting position to a given finishing position. We believe that the placement of obstacles allows us to construct the desired path. Given the convexity of the obstacles (in terms of cells) and their isolation, a heuristic algorithm for autonomous control of the robot's movement has been constructed, which ensures its movement to a given finishing position. To minimize the number of steps (passed cells), a "least square rule" is proposed. Commands for analyzing the state of the robot and the environment, motion commands, and analytical commands are defined. All commands are presented in the form of methods of the corresponding classes for tasks of autonomous robot motion control. The proposed algorithm is implemented in the form of a software package for modeling and visualization of robot motion.

Key words: robot, navigation, plane, model, path, command, algorithm.