Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розробка системи підтримки прийняття рішень для завдань маніпуляції
(тема)

Виконав:
студент 2 курсу, групи КТРСм-21-1

Катков О.В.
(прізвище, ініціали)

Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)

Тип програми Освітньо-професійна

Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник проф. Цимбал О.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та мехатроніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми	освітньо-професійна
Освітня програма	Комп'ютеризовані та робототехнічні системи

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри Невлюдов І.Ш
(підпис)
«_____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Каткову Олексію Валентиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи підтримки прийняття рішень для завдань маніпуляції
затверджена наказом по університету від 07.11. 2022 р. № 1462 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 21.12. 2022 р.
3. Вихідні дані до роботи задача переміщення маніпулятора, методи та алгоритми розробки СППР.

4. Перелік питань, що потрібно опрацювати в роботі:
4.1 аналіз предметної області та технічного завдання;
4.2 аналіз існуючих підходів побудови маршруту для робота- маніпулятора;
4.3 дослідження алгоритмів пошуку сімейства швидкого дослідження випадкових дерев, імітаційного моделювання, порівняння показників та характеристик алгоритмів;
4.4 створення СППР, експериментальні дослідження.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.pptx) – 10 с.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Керівник (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на виконання кваліфікаційної роботи	15.09.2022	Виконано
2	Визначення актуальності роботи, мети, предмета, об'єкта, та розробка завдань для досягнення мети	22.09 – 30.09.22	Виконано
3	Аналіз літературних джерел	01.10 – 13.10.22	Виконано
4	Вибір середовища розробки	15.10 – 3.11.22	Виконано
5	Оформлення пояснювальної записки та документації	05.11 – 19.11.22	Виконано
6	Оформлення додатків	20.11 – 01.12.2	Виконано
7	Оформлення графічного та презентаційних матеріалів комп'ютерного захисту	01.12 – 06.12.22	Виконано
8	Представлення роботи на рецензування	14.12 – 19.12.22	Виконано

Дата видачі завдання 15.09.2022

Студент _____ Катков О.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Цимбал О.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 141 с., 37 табл., 53 рис., 3 дод., 30 джерел.

АВТОМАТИЗАЦІЯ, РОБОТ-МАНІПУЛЯТОР, СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, ПОШУК ШЛЯХУ В ОБ'ЄМНОМУ СЕРЕДОВИЩІ.

Мета роботи – розробка компонентів програмного забезпечення СППР для завдань маніпуляції.

Об'єкт дослідження – процес керування маніпулятором робота.

Предмет дослідження – програмна реалізація СППР для завдань маніпуляції.

Методи дослідження: теорія алгоритмів, теорія структур даних, імітаційне моделювання, логічний та системний аналіз.

У роботі виконано аналіз предметної області, розглянуто сучасні підходи до побудови маршруту для робота-маніпулятора в об'ємному середовищі, досліджено сімейство алгоритмів швидкого розширення випадкових дерев, проведено імітаційне моделювання та на основі отриманих даних розроблена система підтримки прийняття рішень для завдань побудови маршруту в об'ємному середовищі для робота-маніпулятора.

ABSTRACT

Explanatory note: 141 p., 37 tabl, 53 fig, 3 app, 30 sources.

AUTOMATION, ROBOT MANIPULATOR, DECISION SUPPORT SYSTEM, WAY FINDING IN A COMMON ENVIRONMENT.

The purpose of the work is the development of software components for the manipulation tasks.

The object of research is the manipulator robot control system.

The subject of the study is the software implementation of SPD for manipulation tasks.

Research methods: theory of algorithms, theory of data structures, simulation modeling, logical and system analysis.

The work analyzed the subject area, considered modern approaches to the construction of a route for a manipulator robot in a three-dimensional environment, investigated a family of algorithms for rapid expansion of random trees, carried out simulation simulations, and based on the obtained data, developed a decision support system for route construction tasks in the capacious environment for a robot-manipulator.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

СППР – система підтримки рішень;

RPP (Randomized Potential Planner) – рандомізованого потенційний планувальник;

PRM (Probabilistic roadmap) - ймовірнісний метод побудови дорожніх карт.

RRT (Rapidly exploring Random Trees) – швидко досліджувані випадкові дерева;

SCARA (Selective Compliance Articulated Robot Arm) – автоматичний маніпулятор з вибірковою відповідністю;

RPP (Randomized Potential Planner) – Рандомізований потенційний планувальник.

ВСТУП

В наш час, розвиток робототехніки йде дуже швидко, а завдання керування маніпуляторами стають все більш актуальними та об’ємними, саме тому об’єктом дослідження даної кваліфікаційної роботи є процес керування роботом-маніпулятором.

Побудова маршруту руху робота-маніпулятора є однією з найважливіших дослідницьких проблем робототехніки. Велика кількість проблем у різних галузях промисловості вирішується за допомогою планування траєкторії руху, та застосовується при керуванні роботом-маніпулятором для досягнення певної мети. Планування траєкторії руху робота-маніпулятора не може бути спроектовано заздалегідь, для кожної окремо взятої карти місцевості, тому завжди є потреба у виборі відповідного алгоритму планування траєкторії руху маніпулятора.

Від правильності обраного алгоритму планування маршруту в об’ємному середовищі для робота маніпулятора, залежить його ефективність функціонування, бо від правильності вибору алгоритму залежать такі важливі показники як довжина побудованого шляху в об’ємному просторі для робота-маніпулятора, швидкість роботи алгоритму та кількість споживаної їм пам’яті.

Проблему вибору правильно алгоритму планування шляху можна назвати як проблему вибору в складних умовах для всебічного і об’єктивного аналізу предметної області. Для вирішення такої проблеми пропонується використання СППР.

Для того щоб досягти поставленої мети у вирішенні проблеми вибору правильного алгоритму, потрібно програмно реалізувати систему підтримки прийняття рішень для завдань маніпуляції, в свою чергу це вимагає дослідити існуючі алгоритми пошуку шляху в об’ємному середовищі, провести аналіз технічного завдання і імітаційне моделювання, щоб потім, опираючись на отримані дані, створити систему підтримки прийняття рішень.

Досягнення поставленої мети зі створення СППР дозволить аналізувати інформацію у найшвидший і найефективніший спосіб для прийняття певного рішення відносно якоїсь конкретної задачі, і буде надавати підтримку в інтерактивному режимі, що прискорює та поліпшує прийняття рішень у складних умовах. Це дозволить більш ефективно керувати роботом-маніпулятором, наближуючи його параметри та параметри його робочого середовища до необхідних, що призведе до скорочення витрат ресурсів при розробці та подальшій експлуатації маніпулятора, саме тому обрана тема є актуальною.

Мета роботи – розробка компонентів програмного забезпечення СППР для завдань маніпуляції.

Об’єкт дослідження – процес керування маніпулятором робота.

Предмет дослідження – програмна реалізація СППР для завдань маніпуляції.

Методи дослідження: теорія алгоритмів, теорія структур даних, імітаційне моделювання, логічний та системний аналіз.

Для досягнення мети планується розв’язати наступні завдання:

– провести аналіз технічного завдання та методів підтримки прийняття рішень;

– дослідити алгоритми пошуку шляху в об’ємному просторі для робота-маніпулятора;

– дослідити застосування методи та алгоритми підтримки прийняття рішень;

– провести експериментальні дослідження;

– виходячи з отриманих даних, створити СППР;

– розробити програмне забезпечення СППР.

Кваліфікаційну роботу виконано згідно [1], [2].

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Маніпулятори роботів та їх застосування

У робототехніці маніпулятор – це пристрій, який використовується для маніпулювання матеріалами без прямого фізичного контакту оператора. Спочатку маніпулятори розроблялися для роботи з радіоактивними або біологічно небезпечними матеріалами, або їх використовували у важкодоступних місцях. Зараз вони використовуються в різноманітних сферах застосування, включаючи автоматизацію зварювання , роботизовану хірургію, космос та ін.

Маніпулятор – це механізм, схожий на руку, який складається з ряду сегментів, зазвичай ковзаючих або з’єднаних між собою, які мають кілька ступенів свободи і призначені для захоплення та переміщення об’єктів.

У промисловій сфері маніпулятор – це допоміжний пристрій для підйому, який використовується, щоб допомогти працівникам підіймати, переміщувати та розміщувати предмети, які є занадто важкими, гарячими, великими або іншим чином занадто складними для роботи одному, або кільком працівникам вручну. На відміну від простих засобів вертикального підйому (крани, підйомники тощо), маніпулятори мають можливість діставатися до вузьких місць [3]. Хорошим прикладом може бути видалення великих штампованих деталей із преса та розміщення їх у стійці чи подібному кріпленні. У зварюванні, наприклад, маніпулятор використовується для збільшення швидкості наплавлення, зменшення людських помилок та інших витрат у виробничих умовах. Крім того, інструменти маніпулятора дають підйомнику можливість нахиляти, котити або обертати деталь для відповідного розміщення. Прикладом може бути зняття деталі з преса в горизонтальному положенні, а потім висування

її для вертикального розміщення в стійці або перекочування деталі, щоб відкрити задню частину деталі.

1.2 Різновиди маніпуляторів

Маніпулятори поділяються на кілька типів за комбінацією з'єднань[4]:

– маніпулятор з декартовою геометрією – ця рука використовує призматичні з’єднання для досягнення будь-якого положення в прямокутному робочому просторі за допомогою декартових рухів ланок;

– маніпулятор з циліндричною геометрією – це рука, плече якої утворено шляхом заміни поясного з’єднання декартового плеча поворотним з’єднанням, його можна розширити до будь-якої точки в межах циліндричного робочого простору за допомогою комбінації переміщення та обертання;

– маніпулятор з плечем полярної (сферичної) геометрії – рука, плече якої формується коли плечовий суглоб декартового плеча замінюється поворотним суглобом;

– маніпулятор, яка має шарнірний (обертовий) важіль із геометрією;

– автоматичний маніпулятор з вибірковою відповідністю (SCARA) – ця рука має два поворотних шарніри в горизонтальній площині, які дозволяють руці розтягуватися в горизонтальному плоскому робочому просторі.

1.2.1 Маніпулятори із декартовою геометрією

Маніпулятор з декартовою геометрією (також званий лінійним роботом) (рис. 1.1) – це промисловий робот, у якого три головні осі керування є лінійними (тобто вони рухаються по прямій лінії, а не обертаються) і розташовані під прямим кутом одна до одної. Ковзні шарніри відповідають руху зап'ястя вгору-вниз, вліво-вправо, вперед-назад. Перевагами такого механічного розташування

є спрощення рішень для керування роботом. Має високу надійність і точність при роботі в тривимірному просторі [5].

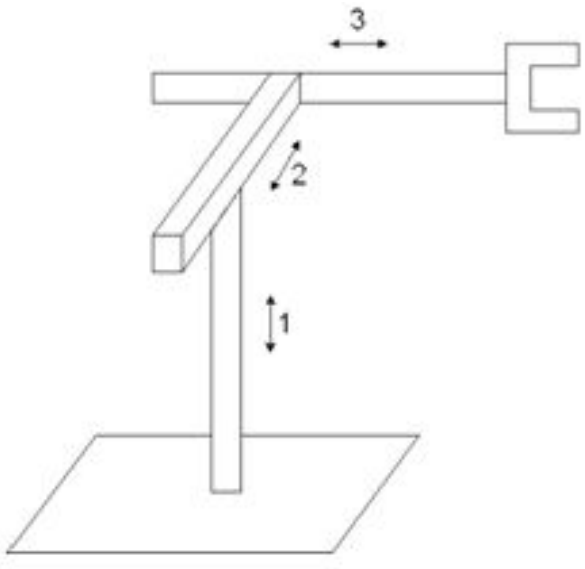

Рисунок 1.1 – Приклад маніпулятора із декартовою геометрією

Подібні роботи мають механізми, що складаються з жорстких ланок, з’єднаних між собою шарнірами з лінійним або обертальним рухом, або комбінацією обох. Типовими прикладами маніпуляторів із декартовою геометрією є стаціонарні лазери, або 3д принтери (рис. 1.2).

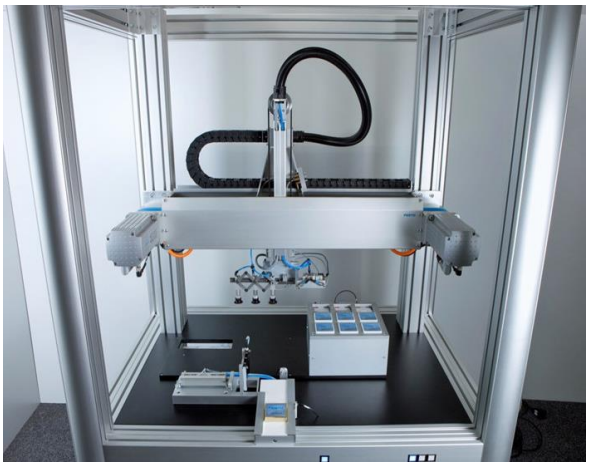

Рисунок 1.2 – Приклад реального маніпулятора із декартовою геометрією

1.2.2 Маніпулятори із циліндричною геометрією

Маніпулятори із циліндричною геометрією (рис. 1.3) мають принаймні одне обертове з’єднання в основі та два лінійних з’єднання. Ця конструкція веде до робочого простору циліндричної форми. Вони зазвичай призначені для роботи у вузьких робочих місцях і ідеально підходять для об’єктів, які повинні мати кругову симетрію, ідеально підходять для шліфування, складання та точкового зварювання.

Такі маніпулятори використовують тривимірну систему координат із обраною базовою віссю та відносною відстанню від неї для визначення положення точки. Відстань до вибраного базового позиціонування та відносний напрямок осей, а також вертикальна відстань до осі від обраної базової площини, також часто використовуються для визначення розташування точки.

Початком системи є точка, у якій усі три координати можна записати як «0», і це точка, де стикаються базова площина та осі. Радіальна відстань - це відстань від осі.

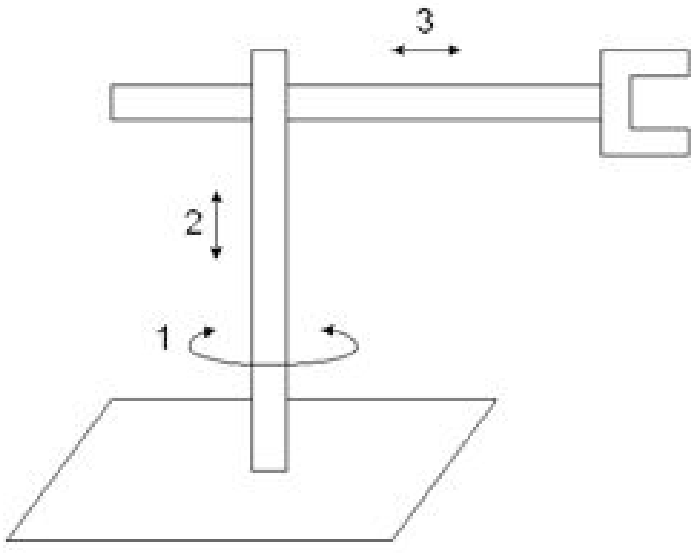

Рисунок 1.3 - Приклад маніпулятора із циліндричною геометрією

1.2.3 Маніпулятори з плечем полярної геометрії

Маніпулятори з плечем полярної геометрії (рис. 1.4), далі полярні маніпулятори , або сферичні маніпулятори, мають руку з двома поворотними шарнірами та одним лінійним шарніром, з'єднаними з основою за допомогою поворотного шарніра. Осі робота працюють разом, утворюючи полярну координату, що дозволяє роботу мати сферичну робочу зону [6].

Полярні маніпулятори вважаються одними з перших типів промислових роботів, які коли-небудь були розроблені. Вони зазвичай використовуються для лиття під тиском, зварювання та обробки матеріалів. Таких роботів також можна використовувати для фарбування, дугового та точкового зварювання. Вони можуть мати великий радіус дії, якщо оснащені лінійною рукою відповідного розміру.

Полярні роботи є застарілою технологією, і їх можна замінити роботами з шарнірною рукою (які також мають сферичну робочу зону), хоча в деяких застосуваннях полярний робот все ще може бути економічно ефективнішим, ніж його альтернативи.

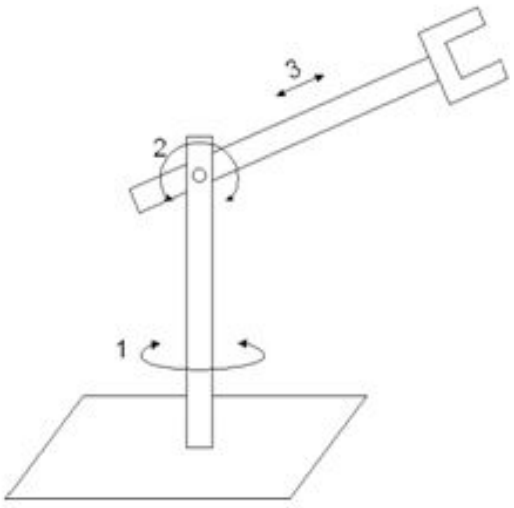

Рисунок 1.4 - Приклад маніпулятора з плечем полярної геометрії

У математиці полярна система координат – це двовимірна система координат, у якій кожна точка на площині визначається відстанню від точки відліку та кутом до напрямку відліку. Сферична система координат, на відміну від полярної – це система координат для тривимірного простору, де положення точки визначається трьома числами: радіальною відстанню цієї точки від фіксованого початку, її полярним кутом, виміряним від фіксованого напрямку в зеніт, і азимутальний кут його ортогональної проекції на площину відліку, яка проходить через початок координат і ортогональна до зеніту, виміряний від фіксованого напрямку відліку на цій площині. Її можна розглядати як тривимірну версію полярної системи координат.

Радіальна відстань також називається радіусом або радіальною координатою. Полярний кут можна назвати зенітним кутом, нормальним кутом або кутом нахилу. Коли радіус фіксований, дві кутові координати утворюють систему координат на сфері, яку іноді називають сферичними полярними координатами.

1.2.4 Шарнірні маніпулятори

Механічний рух і конфігурація шарнірних роботів (рис. 1.5) дуже нагадують людську руку. Рука кріпиться до основи за допомогою поворотного з’єднання. Сама рука може містити від двох обертових суглобів до десяти поворотних суглобів, які діють як осі, причому кожен додатковий суглоб або вісь забезпечує більший ступінь руху. Більшість шарнірних маніпуляторів використовують чотири або шість осей [7].

Шарнірні маніпулятори дуже гнучкі, оскільки всі їхні суглоби можуть обертатися, на відміну від лінійних або поступальних суглобів, які можуть рухатися лише по прямій лінії. Такі маніпулятори використовуються для підбирання та розміщення, дугового зварювання, точкового зварювання, пакування, догляду за машиною та транспортуванням матеріалів. Здатність

створювати дуги (або подібні візерунки) у важкодоступних місцях робить шарнірних роботів хорошим кандидатом для автомобільної промисловості та інших виробництв, де ця здатність потрібна.

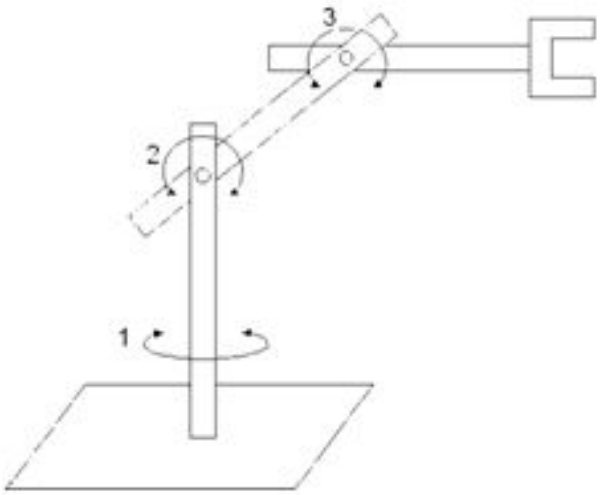

Рисунок 1.5 – Приклад шарнірного маніпулятора

Потужні шарнірні маніпулятори великого розміру здатні брати об’єкти вагою в тонну або більше і людина яка ненароком стане на шляху такого робота, може отримати тяжкі травми (рис. 1.6). Саме тому, маніпулятори з шарнірною рукою можуть оснащуватися відповідними датчиками та керуючим програмним забезпеченням, вона може безпечно працювати поблизу людей. Наприклад, використовуючи комп’ютерне бачення з камерою або, можливо, датчиком LiDAR, якщо такий робот відчуває, що щось є на шляху його руху, або навіть, якщо людина чи об’єкт наближається, але ще не на шляху, робот може уповільнити або зупинити свій рух. Інші датчики можуть виявляти зіткнення – якщо рука натрапляє на щось, вона зупиняється та/або змінює курс. До того ж зазвичай шарнірні маніпулятори, які працюють поруч з людьми, сконструйовані з обмеженнями щодо їх швидкості та сили.

Рисунок 1.6 – Приклад реального шарнірного маніпулятора

1.2.5 Автоматичні маніпулятори з вибірковою відповідністю (SCARA)

SCARA – це абревіатура, що означає Selective Compliance Articulated Robot Arm, або шарнірні маніпулятори з вибірковою відповідністю (рис. 1.7). Маніпулятори SCARA схожі на декартові маніпулятори тим, що вони рухаються за трьома шарнірами або осями. Однак, на відміну від декартових маніпуляторів, у маніпуляторів SCARA два шарніри є обертовими. Тому вони здатні до більш складних рухів, ніж декартові маніпулятори. Зазвичай вони швидші та мають більшу гнучкість у русі, але менш точні, ніж декартові [8]. Як правило, маніпулятори SCARA використовуються для складання та палетування, а також для біомедичних застосувань (рис. 1.8).

Вибіркова відповідність є досить корисною під час монтажних операцій, коли, наприклад, деталь потрібно точно вставити в отвір плати. Центрування деталі в отворі часто потребує невеликої «податливості», щоб знайти центральну точку – трохи поворушити. Після чого вставна сила, що рухає об’єкт вниз, має бути твердою та жорсткою.

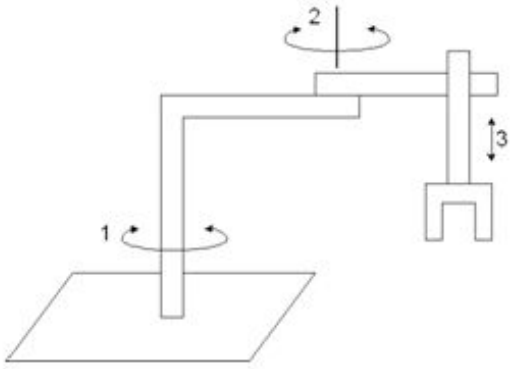

Рисунок 1.7 – Приклад маніпулятора SCARA

Маніпулятори SCARA менш жорсткі, ніж, наприклад, декартові маніпулятори, і також обмежені вагою, яку вони можуть підняти, порівняно з декартовим маніпулятором, який може підіймати досить важкі предмети. Ці маніпулятори займають невелику площу та зазвичай використовуються для додатків, у яких відстань, яку потрібно переміщати, відносно мала. Вони мають менше ступенів свободи, ніж шарнірні руки. З іншого боку, вони мають високу точність. І вони дешевші, ніж шарнірна рука.

Робоча зона маніпулятора SCARA має циліндричну форму. Маніпулятори цього типу, як правило, швидше, ніж шарнірні руки, хоча й не такі швидкі, як роботи Delta.

Рисунок 1.8 – Приклад реального маніпулятора SCARA

1.3 Планування шляху

Автономні роботи-маніпулятори характеризуються здатністю виконувати завдання без будь-якого втручання людини. Прийняття рішень вимагає повного або часткового знання навколишнього середовища або робочого простору, в якому працює маніпулятор. Невизначеність на етапі сприйняття призводить до накопичення помилок локалізації, тому обробка зібраних даних і врахування помилок є важливими для точного картографування та локалізації [9].

Етап планування передбачає розробку стратегії без зіткнень від поточного розташування маніпулятора до бажаного розташування переносимого об’єкта. Планування шляху – це суто геометричний процес, який пов’язаний лише з пошуком шляху без зіткнень, незалежно від можливості реалізації шляху [10]. З іншого боку, кінодинамічне планування враховує кінематику та динаміку робота. Після вказівки шляху кінцевою процедурою є керування рухом або виконання.

Планування є не тільки однією з фундаментальних проблем робототехніки, вона, мабуть, найбільш вивчена. Ранні спроби розробити методи детермінованого планування показали, що їх складно обчислити навіть для простих систем.

Методи точних дорожніх карт, так як графіки видимості, діаграми Вороного, тріангуляція Делоне, адаптивні дорожні карти, намагаються зафіксувати зв’язок робочого простору робота.

Методи клітинної декомпозиції, в яких робочий простір поділяється на невеликі клітини, були застосовані в робототехніці. Алгоритми пошуку, такі як алгоритм Дейкстри [11] , знаходять оптимальне рішення в графі зв’язності, тоді як D* [12] і A* [13] адаптовані до динамічних графів. Використання методів пошуку графів передбачає дискретизацію робочого простору, і їхня продуктивність погіршується у великих розмірах.

Поява нових обчислювальних методів, надихнула на їх використання в плануванні шляху. Такі методи, як нечітке логічне керування, нейронні мережі, генетичні алгоритми, оптимізація мурашиної колонії, усі почали застосовувалися в плануванні шляху робота. Було запропоновано реактивні методи планування на основі датчиків, але їх не можна вважати глобальними планувальниками, бо методи, засновані на контролі, вимагають формулювання точних моделей робота та середовища, що може бути досить важким завданням.

1.4 Планування на основі вибірки

Планування на основі вибірки (Sampling-Based Planning) унікальне тим фактом, що планування відбувається шляхом вибірки з простору конфігурації. У певному сенсі SBP намагається зафіксувати зв’язність простору конфігурації шляхом його вибірки [14]. Цей довільний підхід має свої переваги з точки зору надання швидких рішень для складних проблем. Недоліком є те, що рішення вважаються неоптимальними. Планувальники, на основі вибірки, не гарантовано знайдуть рішення, якщо воно існує. Рішення буде надано, якщо воно існує, за умови достатнього часу виконання алгоритму (у деяких випадках нескінченного часу виконання).

Планування на основі вибірки аж ніяк не є новою концепцією в робототехніці [15]. Було запропоновано подолати складність детермінованих алгоритмів планування для робота з шістьма ступенями свободи. Використання випадкових обчислень для вирішення досить складних проблем було надзвичайно успішним і надихнуло на розробку рандомізованого потенційного планувальника (Randomized Potential Planner). RPP використовував випадкові блукання, щоб уникнути локальних мінімумів потенційного планувальника поля. Пізніше було запропоновано планувальник, повністю заснований на випадкових блуканнях, з адаптивними параметрами.

Робота Джеремі Барраканда та Жана-Клоде Латомбе проклала шлях до нового покоління алгоритмів планування руху, які використовують рандомізацію. Зараз, мабуть, найпоширенішими алгоритмами є ймовірнісний метод дорожніх карт (Probabilistic roadmap) [16] і алгоритм швидкого дослідження випадкових дерев (Rapidly-exploring Random Trees) [17]. Одночасно було розроблено кілька інших алгоритмів, які перевершили RPP. Інтуїтивно зрозуміле впровадження як RRT, так і PRM, а також якість рішень призвели до їх широкого застосування в робототехніці та багатьох інших галузях.

Алгоритм ймовірнісної дорожньої карти (рис. 1.9) реалізує дві основні процедури для створення імовірнісної дорожньої карти.

Спочатку відбувається фаза навчання, коли простір конфігурації відбирається протягом певного часу. Конфігурації у вільному просторі зберігаються, тоді як ті, що знаходяться в просторі перешкод, відкидаються.

Далі слідує фаза запиту, де початкова та цільова конфігурації визначаються та пов’язуються з дорожньою картою. Дорожні карти іноді називають лісами, як аналогію з деревами в RRT.

У результаті підтримки дорожньої карти та визначення початкової та цільової конфігурацій на наступному етапі PRM може вирішувати різні проблеми в одному середовищі. Його називають багатозапитовим планувальником [16]. Час планування витрачається на вибірку та створення дорожньої карти, щоб запити вирішувалися швидко.

RRT представляє іншу категорію планувальників на основі вибірки, які є планувальниками з одним запитом. Дерево поступово розростається від початкової конфігурації до цільової або навпаки (рис. 1.10). Конфігурація вибирається випадковим чином у просторі конфігурацій [19]. Якщо він лежить у вільному просторі, робиться спроба підключення до найближчої вершини дерева. Для вирішення одного запиту RRT є швидшим порівняно з PRM. Не потрібно брати вибірку простору конфігурації та будувати дорожню карту, тобто пропускається фаза навчання.

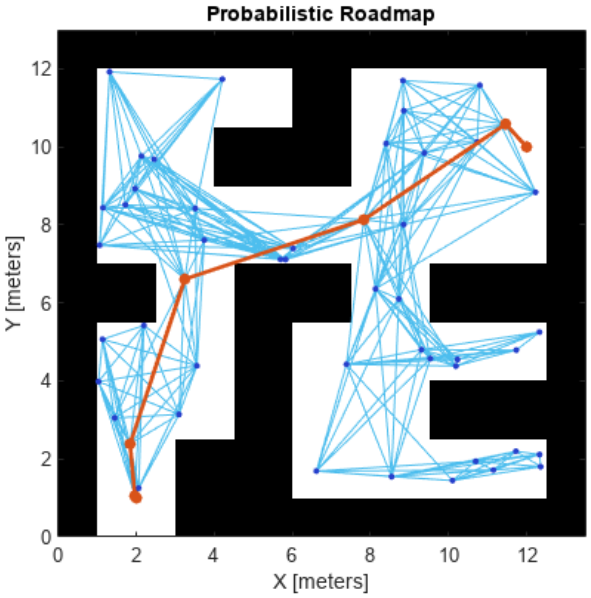

Рисунок 1.9 – Приклад роботи PRM алгоритму

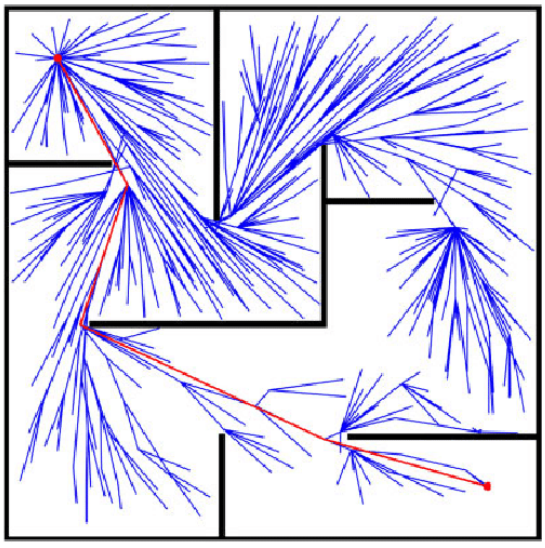

Рисунок 1.10 – Приклад роботи RRT алгоритму

1.5 Системи підтримки прийняття рішень

Система підтримки прийняття рішень (СППР) – комп'ютерна автоматизована система, метою якої є допомога людям у прийнятті рішення в складних умовах, для повного та об'єктивного аналізу предметної діяльності.

Концепція СППР виникла в кінці 60-х років XX століття разом з ідеєю розподіленого комп'ютерного обчислення. Першою метою створення таких систем було надання кінцевим користувачам можливості взаємодіяти безпосередньо з комп'ютером без посередництва інформаційних спеціалістів, та подолання потреби у створенні відповідних комп'ютерних додатків для розроблення управлінських рішень [20].

Система підтримки прийняття рішень (СППР) призначена для підтримки багатокритеріальних рішень у складному інформаційному середовищі. При цьому під багатокритеріальністю розуміється, що результати прийнятих рішень оцінюються не по одному, а за сукупністю багатьох показників (критеріїв) аналізованих одночасно. Інформаційна складність визначається необхідністю обліку великого обсягу даних, обробка яких без допомоги сучасної обчислювальної техніки практично неможлива. У цих умовах число можливих рішень, як правило, дуже велике, і вибір найкращого з них людиною-експертом, з посиланням на її власне бачення, без всебічного аналізу, може призводити до грубих помилок.

Система підтримки рішень СППР вирішує два основні завдання. По-перше, вибір найкращого рішення з багатьох можливих (оптимізація). По-друге, упорядкування можливих рішень за перевагою (ранжування).

В обох завданнях першим і найважливішим моментом є вибір сукупності критеріїв, на основі яких надалі оцінюватимуться та зіставлятимуться можливі рішення (назвемо їх також альтернативами). Система СППР допомагає користувачеві зробити такий вибір. –

Для аналізу та вироблення пропозицій у СППР використовуються різні методи. Це можуть бути:

– інформаційний пошук;

– інтелектуальний аналіз даних;

– пошук знань у базах даних;

– міркування на основі прецедентів;

– імітаційне моделювання;

– еволюційні обчислення та генетичні алгоритми;

– нейронні мережі;

– ситуаційний аналіз;

– когнітивне моделювання та ін.

Деякі з цих методів були розроблені у рамках штучного інтелекту. Якщо в основі роботи СППР лежать методи штучного інтелекту, то говорять про інтелектуальну СППР або ІСППР.

Система дозволяє вирішувати завдання оперативного та стратегічного управління на основі облікових даних про діяльність компанії.

Система підтримки прийняття рішень є комплексом програмних інструментальних засобів для аналізу даних, моделювання, прогнозування та прийняття управлінських рішень, що складається з власних розробок та готових програмних продуктів.

1.6 Структура СППР

Система підтримки прийняття рішень є інтерактивною системою, яка забезпечує користувачеві легкий доступ до моделей і даних для того, щоб підтримати процес прийняття рішень стосовно слабоструктурованих і неструктурованих завдань.

СППР характеризуються чіткою структурою (рис. 1.11), яка містить чотири головні компоненти:

– модель;

– підсистема аналізу;

– базу даних;

– інтерфейс користувача.

Модель – це формальний опис системи. При цьому сама система може бути якого завгодно призначення: економічною, технологічною, соціальною та ін. В процесі розробки до модель подають певні дані:

– технічно технологічні;

– економічні;

– соціальні.

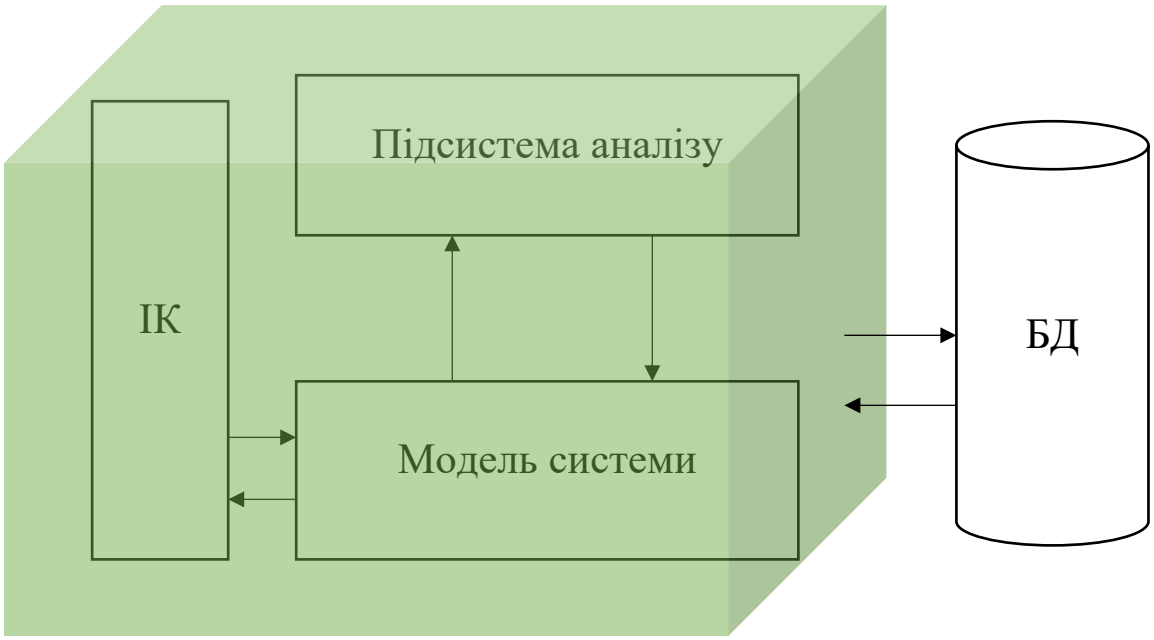

Рисунок 1.11 – Структурна схема СППР

Крім цього, часто модель спирається на накопичену статистику, використовуючи власну базу даних, або долучається до кооперативної.

Підсистема аналізу – це набір інструментів, для проведення аналізу моделі. Часто застосовують наступні типи експериментів:

– «що буде, якщо?» – експеримент дозволяє запустити модель з різними параметрами;

– «оптимізація» - експеримент, що дозволяє знаходити оптимальні параметри моделі;

– «аналіз чуттєвості» - експеримент, що використовується для оцінки впливу одного з параметрів на результат моделювання;

– «метод Монте-Карло» - експеримент, що дозволяє оцінити результати стохастичних моделей.

Також СППР включає в себе інтерфейс користувача – комп'ютерну оболонку, яка дозволяє запускати обчислювальні експерименти з моделлю та дивитися отримані результати

1.7 Моделі СППР

В повсякденній діяльності для допомоги у прийнятті рішень використовують дві моделі СППР: аналітичну модель, та «динамічну» модель.

При аналітичному моделюванні процеси функціонування досліджуваної системи записують як рівняння і логічні співвідношення. Аналіз моделі при цьому зводиться до їхнього аналітичного рішення.

Аналітичні моделі допомагають знайти деякі покращення в роботі і не потребують спеціальних систем розробки. Тобто може використовуватися стандартна модель на базі Exel. Але такі моделі не враховують швидкість роботи системи да час на виконання поставленої задачі, не враховують фактори випадковості, складні причинні взаємозв'язки або складні обмеження.

На відміну від аналітичних моделей, «динамічні» потребують спеціальних інструментів розробки, але дозволяють враховувати часові і причинні взаємозв'язки та обмеження будь-якої складності, враховують фактори випадковості, показують детальну поведінку системи у часі та дозволяють показати і виміряти майже всі дані стосовно роботи системи.

1.8 Вибір альтернативи

Головним питанням у проведенні процедури вибору альтернативи є критерій такого вибору, відповідно до якого, задаються пріоритети у прийнятті рішень. До характерних, для оптимального вибору альтернативного рішення, критеріїв можна віднести: надійність, технологічність, оперативність, економічність, якість, продуктивність, корисність та ін.

Серед найбільш відомих методів, що дозволяють здійснювати ефективний вибір альтернатив у прийнятті рішень, слід зазначити:

– метод «згортки», у якому розраховуються значення єдиного комплексного критерію кожного альтернативного варіанта розв'язку;

– принцип Парето, у якому зіставляються оцінки альтернативних варіантів рішень за кількома критеріями і відкидаються «доміновані» рішення;

– лексикографічний вибір, при якому вибір здійснюється спочатку за найважливішими критеріями, а потім за менш важливими.

У рамках даної роботи було обрано наступні критерії для алгоритмів пошуку шляху о об'ємному середовищі:

– швидкість роботи алгоритму;

– довжина побудованого маршруту;

– споживання алгоритмом пам'яті.

1.9 Висновки до першого розділу

Провівши ґрунтовний аналіз наукових досліджень стосовно алгоритмів заснованих на sampling-based плануванні, а саме PRM та RRT алгоритмів, і посилаючись на наведені в цьому розділі відомості стосовно алгоритмів, не пов'язаних з sampling-based плануванням, можна зробити висновок що алгоритм швидкого дослідження випадкових дерев (RRT) у випадках обробки одного єдиного запиту, виходячи з адаптивності до різних видів маніпуляторів,

швидкості роботи та універсальності у застосуванні в двовимірних та тривимірних просторах, є найкращим алгоритмом планування шляху.

У рамках дипломної роботи буде проведено аналіз сімейства алгоритмів RRT, в залежності від параметрів початкового простору конфігурації.

Алгоритми будуть порівняні за такими критеріями як:

– швидкість роботи алгоритму;

– довжина побудованого маршруту;

– споживання алгоритмом пам’яті.

У якості метода прийняття рішень для розроблюваної СППР було обрано метод «згортки».

2 ДОСЛІДЖЕННЯ АЛГОРИТМІВ ПОШУКУ ШЛЯХУ НА ОСНОВІ ВИБІРКИ

2.1 Алгоритм швидкого дослідження випадкових дерев (RRT)

Проблеми планування руху виникають у таких різноманітних областях, як робототехніка, створення віртуальних прототипів, виробництво та комп'ютерна анімація. Такі проблеми передбачають пошук у просторі конфігурації системи одного або кількох складних геометричних тіл для шляху без зіткнень, який з'єднує дану початкову та цільову конфігурації, задовольняючи при цьому обмеження, накладені складними перешкодами. Хоча повні алгоритми відомі для цього загального класу задач, їх обчислювальна складність обмежує їх використання просторами конфігурації низької розмірності. Це обмеження, результати експериментів з нижньою межею жорсткості та сильна мотивація вирішувати практичні проблеми планування стимулювали розвиток та успіх багатьох методів планування шляху, які використовують рандомізацію. Прийнятим компромісом є те, що методи є неповними, але вони знайдуть рішення з будь-якою ймовірністю за достатнього часу роботи. Головне – розробити рандомізовані методи, які швидко сходяться на практиці, але є достатньо простими, щоб забезпечити узгоджену поведінку та аналіз.

Рандомізовані алгоритми планування шляху зазвичай розробляються для одного з двох контекстів: планування з одним запитом і планування з кількома запитами. Для планування з одним запитом передбачається, що проблема планування з одним шляхом повинна бути вирішена швидко, без будь-якої попередньої обробки. Одним із найперших і найпопулярніших методів вирішення цієї проблеми був підхід рандомізованого потенційного поля [21]. Для планування з кількома запитами передбачається, що багато проблем планування шляху будуть вирішені для одного середовища. У цьому випадку варто попередньо обробити інформацію та зберегти її в структурі даних, яка

дозволяє швидко планувати запити. Підхід імовірнісної дорожньої карти був першим для вирішення цієї проблеми, а на його змінну прийшов алгоритм швидкого дослідження випадкових дерев. Граф будується в просторі конфігурації шляхом випадкового вибору багатьох конфігурацій і використання локального планувальника для з'єднання пар сусідніх конфігурацій.

Завдяки своїй простоті та надійності підхід швидкого дослідження випадкових дерев користується значним успіхом в останні роки, і поточні дослідження зосереджені на аналізі та лікуванні патологічних випадків. Навіть для задач з одним запитом, де підхід рандомізованого потенційного поля може дати кращу продуктивність, метод швидкого дослідження випадкових дерев був кращим через його надійність [22].

Планування шляху загалом можна розглядати як пошук у просторі конфігурації C, у якому кожен $q \in C$ визначає положення та орієнтацію одного чи кількох геометрично складних тіл у 2D чи 3D вимірі. Вважається, що метрика ρ визначена на C. Нехай C_{free} позначає набір конфігурацій, для яких ці тіла не стикаються з будь-якими статичними перешкодами. Перешкоди повністю моделюються у світі, і явне представлення C_{free} недоступне. Однак, використовуючи алгоритм виявлення зіткнень, задану множину $q \in C$ можна перевірити, щоб визначити, чи $q \in C_{free}$.

Завдання планування шляху одного запиту полягає в обчисленні безперервного шляху від початкової конфігурації, q_{init} до цільової конфігурації q_{goal}, без виконання будь-якої попередньої обробки. Швидко досліджуване випадкове дерево (RRT) було представлено як ефективна структура даних і схема вибірки для швидкого пошуку у просторах великої розмірності, які мають як алгебраїчні обмеження (що виникають через перешкоди), так і диференціальні обмеження (що виникають через неголономію та динаміку). Ключова ідея полягає в тому, щоб спрямувати дослідження на недосліджені частини простору.

Основний алгоритм побудови RRT наведено на рисунку 2.2.

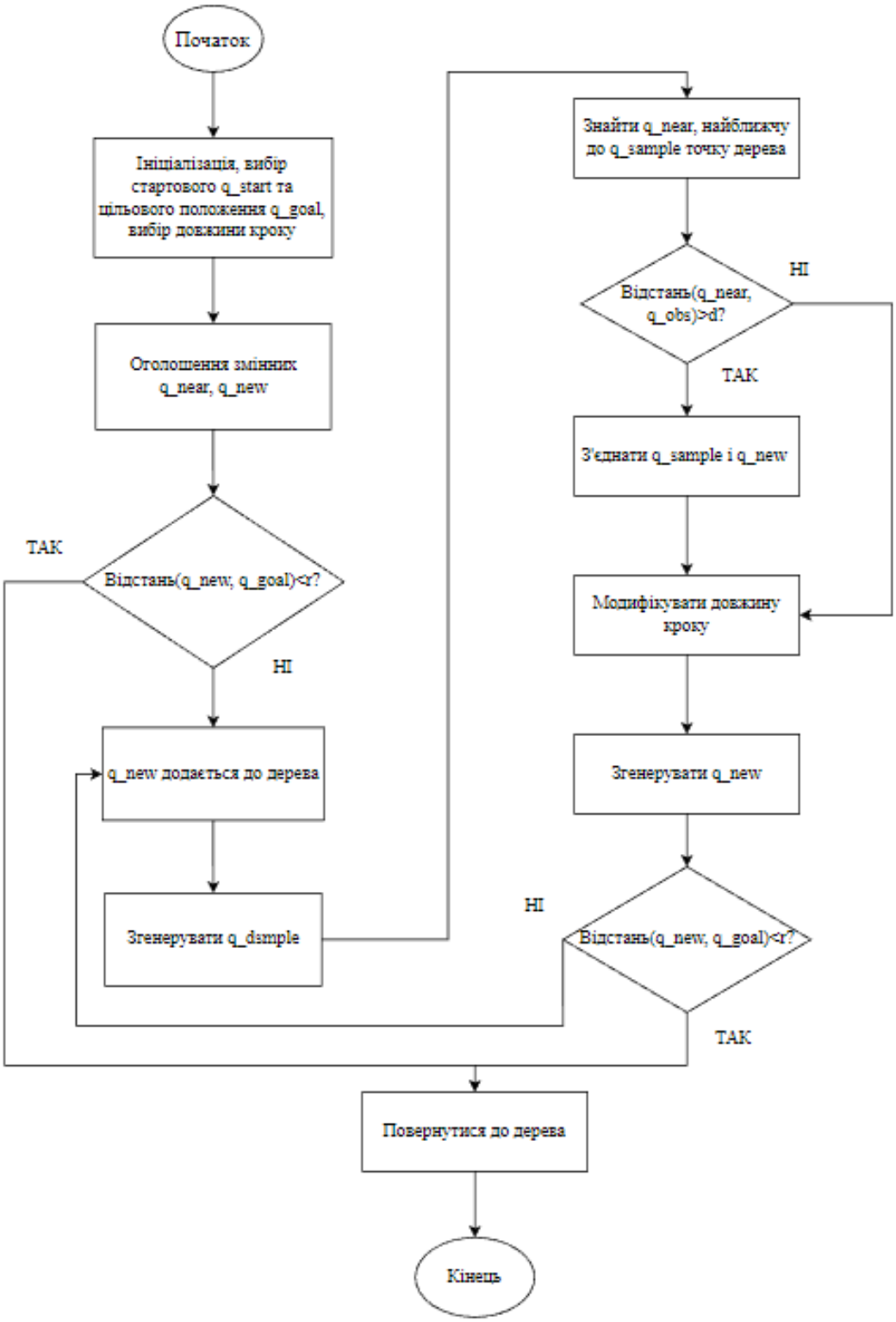

Рисунок 2.2 – Алгоритм побудови RRT

Виконується проста ітерація, у якій кожен крок намагається розширити RRT шляхом додавання нової вершини, яка зміщена випадково вибраною

конфігурацією. Функція EXTEND, яка показана на рисунку 2.3, вибирає найближчу вершину, яка вже є в RRT, до заданої приблизної конфігурації q_{goal}. Функція NEW CONFIG здійснює рух у напрямку q_{goal} з деякою фіксованою дистанцією збільшення та перевіряє зіткнення. Це можна зробити швидко («майже за постійний час») за допомогою алгоритмів інкрементального обчислення відстані.

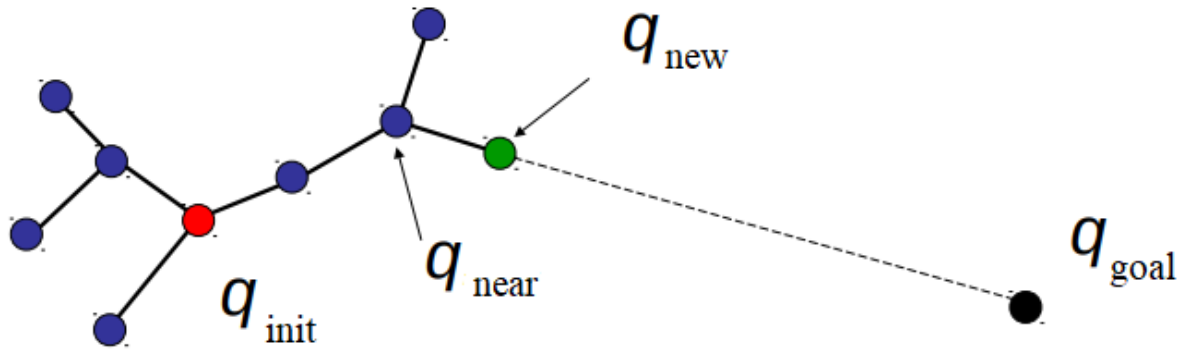

Рисунок 2.3 – Функція EXTEND

Можуть виникнути три ситуації:

– коли цільову конфігурації досягнуто , тобто коли q_{goal} безпосередньо додається до RRT, оскільки дерево вже містить вершину в межах досяжності q_{goal};

– ситуація, коли дерево просунулося на один крок до цільової конфігурації при якій до RRT додається нова вершина $q_{new} \neq q_{goal}$;

– ситуація, коли дерево наштовхнулося на перешкоду, при виконанні якої, запропонована наступна нова вершина відхиляється, оскільки вона не лежить у C_{free}.

На рисунку 2.4 зображено RRT, побудований у квадратному просторі, та діаграму Вороного для вершин RRT.

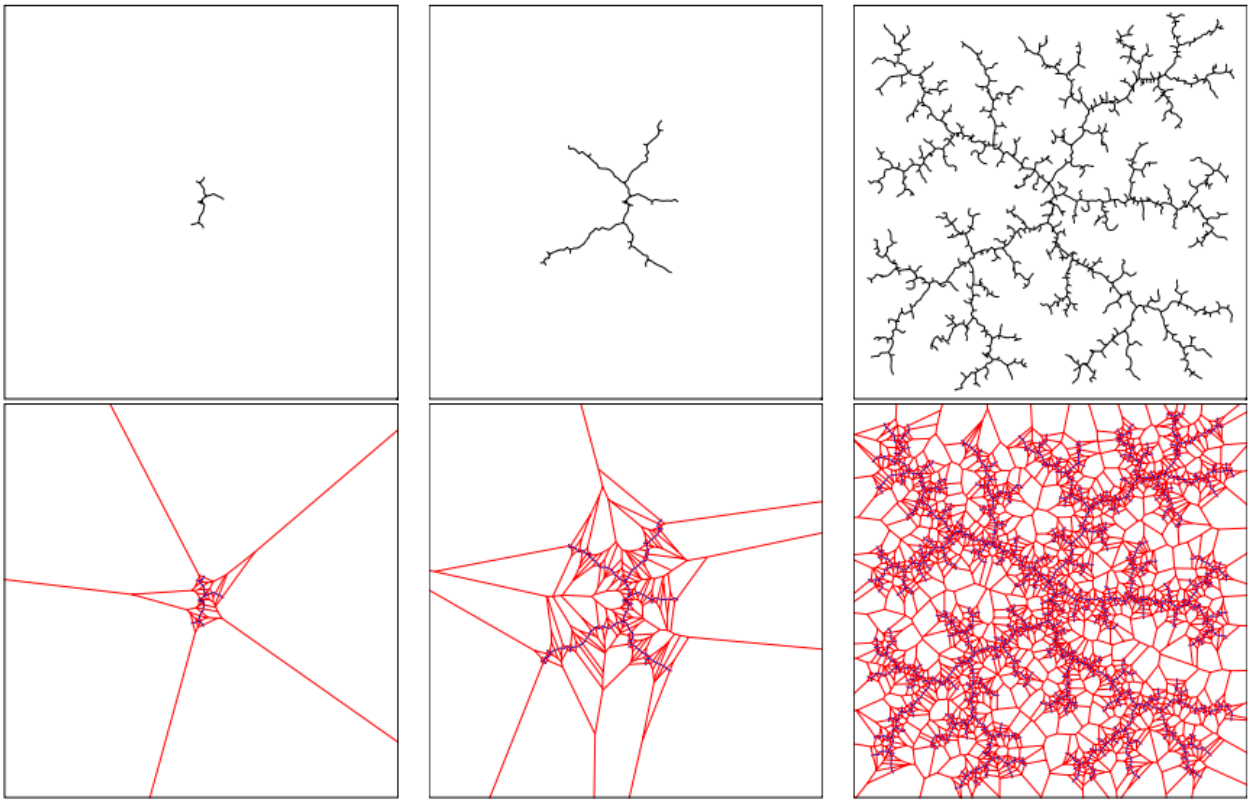

Рисунок 2.4 – RRT зміщується великими регіонами Вороного для швидкого дослідження карти, перш ніж рівномірно охопити простір

Треба зазначити, що ймовірність того, що вершина буде обрана для розширення, пропорційна площі її області Вороного. Це спричиняє те, що RRT схиляється до швидкого дослідження карти. Крім того, треба зазначити, що RRT досягають рівномірного покриття простору, що також є бажаною властивістю ймовірнісного планувальника дорожньої карти.

2.2 Алгоритм RRT*

RRT^* відрізняється від алгоритмів RRT лише тим, що він обробляє процедуру розширення. Сам алгоритм отримано шляхом модифікації RRG.

Алгоритм RRG подібний до RRT тим, що він спочатку намагається підключити найближчий вузол до нового зразка. Якщо спроба підключення вдалася, новий вузол додається до набору вершин [23]. Однак RRG має наступну відмінність. Щоразу, коли нова точка додається до набору вершин V , тоді з'єднання намагаються встановити з усіх інших вершин у V, які знаходяться в межах кулі деякого радіуса r. Для кожного успішного з'єднання до набору ребер E додається нове ребро. Отже, зрозуміло, що для тієї самої послідовності вибірки граф RRT (спрямоване дерево) є підграфом графа RRG (неорієнтованого графа, можливо, що містить цикли).

Але підтримка деревоподібної структури, а не графа, є не тільки економною з точки зору вимог до пам'яті, але також може бути вигідною в деяких програмах, наприклад, завдяки відносно легким розширенням задач планування руху з диференціальними обмеженнями або для усунення помилок моделювання. Тому алгоритм RRT^* отримано шляхом модифікації RRG таким чином, щоб уникнути утворення циклів, шляхом видалення «надлишкових» ребер, тобто ребер, які не є частиною найкоротшого шляху від кореня дерева (тобто початкового стану дерева) до вершини. Оскільки графи RRT і RRT^* є спрямованими деревами з однаковим коренем і набором вершин, а набори ребер є підмножинами графів RRG, це означає «переналаштування» дерева RRT, гарантуючи, що вершини досягаються через мінімальний вартість шляху.

Розглянемо систему з динамікою такого вигляду:

$$x^{\cdot}(t) = f(x(t), u(t)) , \tag{2.1}$$

де $\mathrm{c}(t) \in \mathrm{C}$ та $u(t) \in U$;

$X \subset \mathbb{R}^d$ та $U \subset \mathbb{R}^m$ позначають простір станів і вхідний простір відповідно.

Нехай C_{obs} позначає область перешкод, а C_{free} визначає простір без перешкод. Нарешті, нехай C_{goal} позначає цільову область. Задача планування шляху полягає в тому, щоб знайти керуючий вхід, який дає можливий шлях $\mathrm{c}(t) \in \mathrm{C}_{free}$ для $t \in [0, T]$ від початкового стану $\mathrm{c}(0) = \mathrm{C}_{init}$ до цільової області $\mathrm{c}(T) \in \mathrm{C}_{goal}$, яка підкоряється динаміці системи.

Задача оптимального планування шляху накладає додаткову вимогу, щоб результуючий можливий шлях мінімізував задану функцію вартості $c(x)$, відображаючи кожну нетривіальну допустиму траєкторію як позитивне дійсне число.

У вирішенні задачі оптимального планування руху алгоритм RRT^* будує та підтримує дерево $T = (V, E)$, що складається з набору вершин V станів із C_{free}, з'єднаних спрямованими ребрами E. Спосіб, у який RRT^* генерує це дерево, дуже нагадує спосіб стандартного RRT, з додаванням кількох ключових кроків для досягнення оптимальності. Алгоритм RRT^* використовує набір базових процедур, які описуються в контексті кінодинамічного планування руху, а саме:

– функція вибірки, яка випадково відбирає стан $q_{rand} \in \mathrm{C}_{free}$ із вільної від перешкод області простору станів;

– функція відстані повертає вартість оптимальної траєкторії між двома станами, припускаючи відсутність перешкод. Без диференціальних обмежень це евклідова відстань;

– функція «найближчого сусіда» – для стану $q \in \mathrm{C}$ і дерева $T = (V, E)$, функція «найближчого сусіда» повертає найближчий вузол у дереві з точки зору функції відстані;

– функція «найближчої вершини» - для стану $q \in \mathrm{C}$ і дерева $T = (V, E)$, функція «найближчої вершини» повертає вершини у V, які знаходяться поблизу q;

– функція перевірки зіткнень перевіряє, чи лежить шлях $\mathrm{c} : [0, T] \to \mathrm{C}$ у безперешкодній області простору станів, тобто чи є $\mathrm{c}(t) \in \mathrm{C}_{\text{акуу}}$ для всіх $t \in [0, T]$;

– функція створення вузла, враховуючи поточне дерево $T = (V, E)$, існуючий стан $q_{current} \in V$ і новий стан q_{new}, функція створення вузла додає q_{new} до V і створює нове ребро з $q_{current}$ як зі своїм попередником, яке додається до E. Він призначає вартість для q_{new}, що дорівнює вартості його попередника, плюс вартість $\mathrm{x}(c)$ траєкторії, пов'язаної з новим ребром.

За винятком процесу приєднання існуючого вузла в дереві до нового вузла, RRT^* по суті поводиться ідентично RRT. RRT^* починається з порожнього дерева та додає один вузол, що відповідає початковому стану. Потім він створює та вдосконалює дерево за допомогою набору N ітерацій. Подібно до RRT, RRT^* поступово будує дерево, відбираючи випадковий стан q_{rand} із простору без перешкод і вирішуючи траєкторію c_{new}, яка розширює дерево від найближчого вузла дерева до цієї випадкової точки. Якщо ця траєкторія не стикається з перешкодами, стандартний RRT вставляє новий вузол q_{new} у дерево з $q_{nearest}$ як його батьківського і продовжує наступну ітерацію. Саме тут робота RRT^* відрізняється. Замість того, щоб вибирати найближчий вузол як батьківський, RRT^* розглядає всі вузли в околиці q_{new} і оцінює вартість вибору кожного як батьківського [24].

Цей процес оцінює загальну вартість як додаткову комбінацію вартості, пов'язаної з досягненням потенційного батьківського вузла, і вартості траєкторії для з'єднання. Вузол, який дає найменшу вартість, стає батьківським, а новий

вузол додається до дерева. Процедура перевірки потім перевіряє кожен вузол q_{near} поблизу q_{new}, щоб побачити, чи досягнення q_{near} через q_{new} досягне менших витрат, ніж досягнення цього вузла через його поточного попередника. Якшо це з'єднання зменшує загальну вартість, пов'язану з q_{near}, алгоритм модифікує (з'єднує) дерево, щоб зробити q_{new} батьківським для q_{near} (рис. 2.5). Потім RRT∗ продовжує наступну ітерацію.

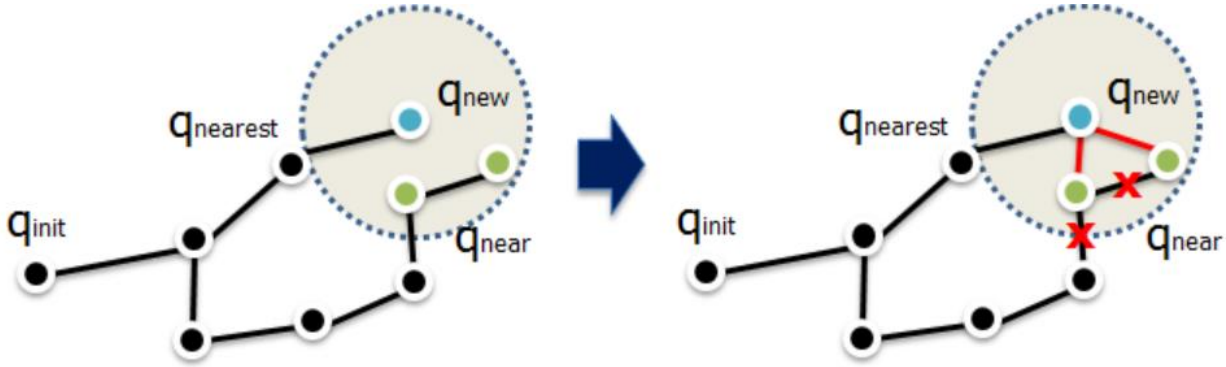

Рисунок 2.5 – Процедура перевірки нових вузлів алгоритмом RRT^*

2.3 Алгоритм RRT-Connect

Оскільки алгоритм RRT генерує вибіркові вузли з однаковою ймовірністю в усьому просторі конфігурації та не враховує розширення в напрямку цільової точки, розширення дерева є дещо сліпим. Для вирішення цієї проблеми був запропонований двонаправлений алгоритм RRT-Connect з ідеєю жадібних розширень. Порівняно з алгоритмом RRT, RRT-Connect має два суттєвих покращення:

– він генерує випадкове дерево з точок початкового та кінцевого стану, і планування завершується, коли два дерева перетинаються, що значно покращує швидкість пошуку алгоритму;

– використовується стратегія розширення вузлів, де на кожній ітерації генерації вузлів алгоритм намагається використовувати найближчий вузол

іншого дерева як напрямок розширення цього дерева, змушуючи два дерева швидко перетинатися.

Крім того, алгоритм продовжить розгортати вузол у цьому напрямку, якщо він не стикається з перешкодою під час розширення в цьому напрямку. Якщо виникне перешкода, алгоритм використовуватиме функцію обміну, яка дозволяє розгортати інше випадкове дерево, що значною мірою запобігає потраплянню алгоритму в дилему локального оптимуму. Використовуючи декілька з цих стратегій, швидкість пошуку та ефективність пошуку RRT-Connect значно підвищилися [25].

Планувальник RRT-Connect розроблено спеціально для проблем планування шляху, які не включають диференціальних обмежень. У цьому випадку потреба в поступових рухах менш важлива. Метод базується на двох ідеях:

– евристика CONNECT, яка намагається переміститися на більшу відстань;

– зростання RRT від q_{init} і q_{goal} одночасно.

Функція CONNECT (рис. 2.6) – це жадібна функція, яку можна розглядати як альтернативу функції EXTEND (рис. 2.3).

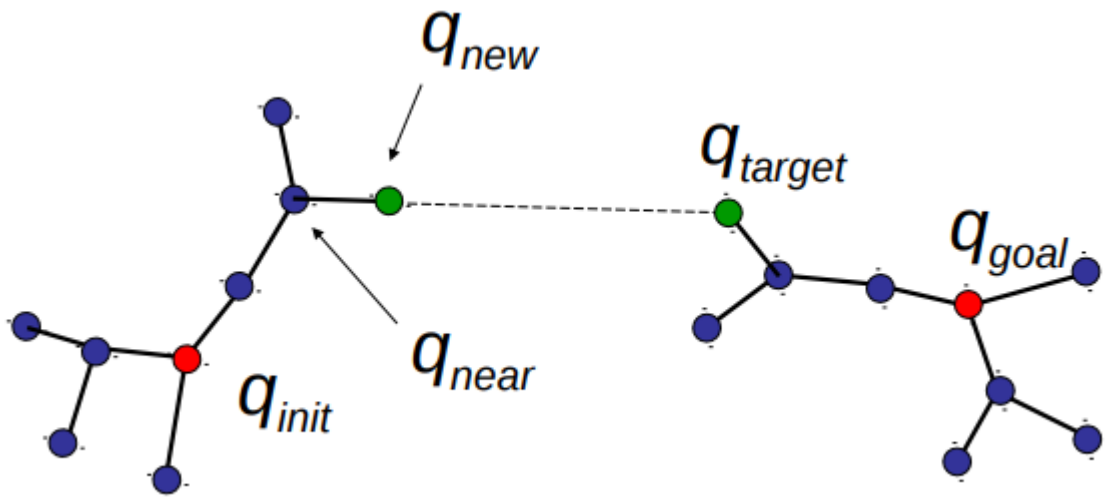

Рисунок 2.6 – Функція CONNECT

Замість того, щоб намагатися розширити RRT одним кроком, функція CONNECT повторює крок EXTEND, доки не буде досягнута q_{goal} (рис. 2.7) або перешкода.

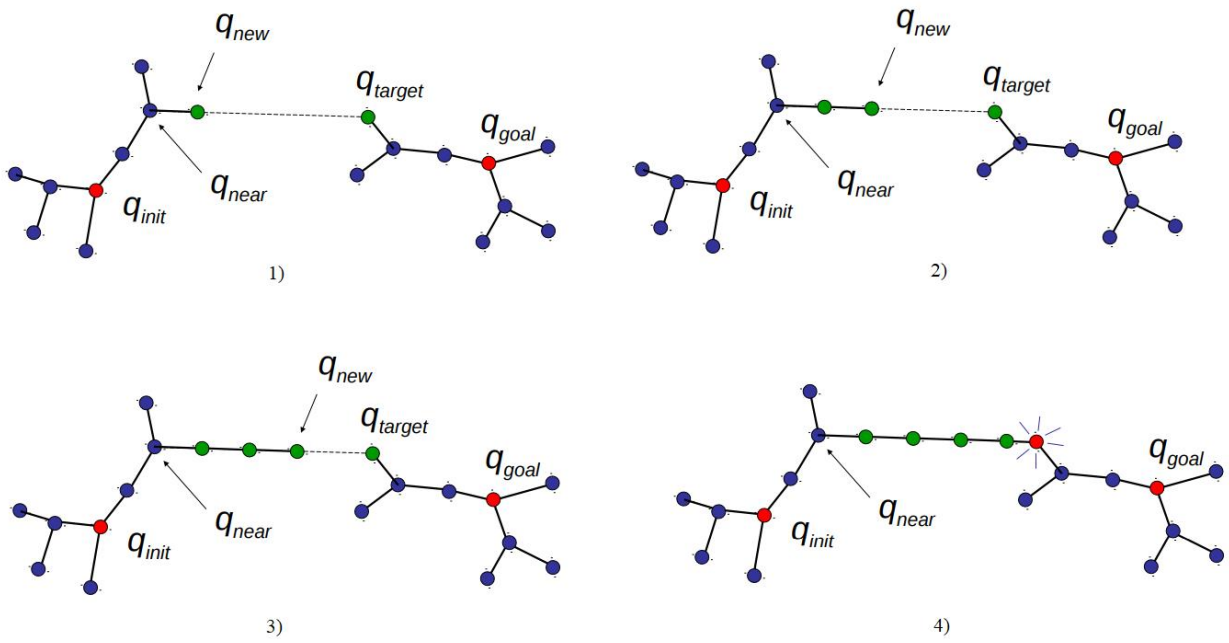

Рисунок 2.7 – Приклад з’єднання двох дерев функцією CONNECT

В обох випадках евристика дозволяє швидко прийти до рішення. У певному сенсі, за допомогою евристики CONNECT область з’єднання продовжує рухатися в міру зростання RRT дерева, на відміну від методу штучного потенційного поля, в якому область з’єднання залишається фіксованою на меті. У алгоритмі RRT-Connect два дерева зберігаються весь час, поки вони не з’єднаються та не буде знайдено рішення (рис. 2.8). У кожній ітерації кожне дерево розширюється, і робиться спроба з’єднати найближчу вершину іншого дерева з новою вершиною. Це змушує обидва дерева досліджувати C_{free}, намагаючись встановити зв’язок між ними.

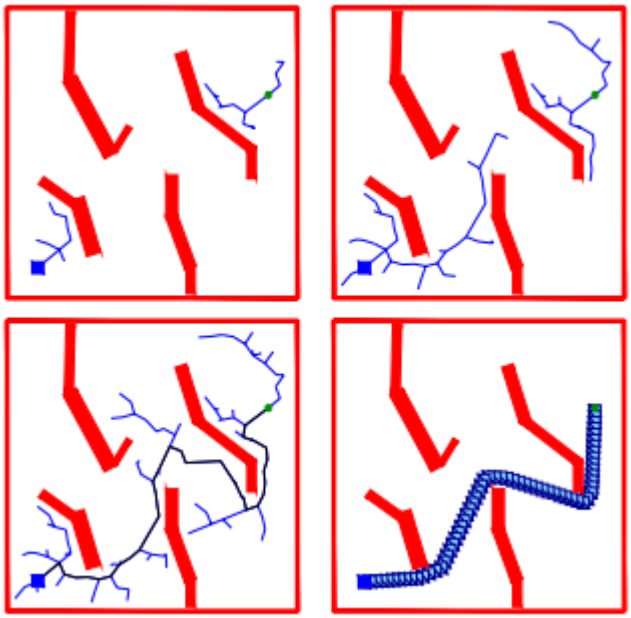

Рисунок 2.8 – Приклад розростання дерева RRT-Connect

Зростання двох RRT дерев також було запропоновано для кінодинамічного планування, однак у кожній ітерації обидва дерева поступово розширювалися до випадкової конфігурації. Поточний алгоритм також намагається вирощувати дерева назустріч одне одному, що, як виявилося, дає набагато кращу продуктивність [26].

Також можна розглянути кілька варіантів вищезазначеного планувальника. Замінивши CONNECT на EXTEND у RRT-Connect, ви отримаєте простий планувальник із двома деревами RRT. Адаптація цього планувальника до проблем, які включають диференціальні обмеження, швидше за все, дасть значне підвищення продуктивності порівняно зі стандартним планувальником RRT.

Інший варіант можна отримати, замінивши EXTEND на CONNECT у RRT-Connect. Це призведе до планувальника шляху з ще сильнішою жадібною евристикою. Однією з ключових переваг функції CONNECT є те, що довгий шлях можна побудувати лише одним викликом алгоритму найближчого сусіда, де кожна нова вершина стане найближчим сусідом для наступної. Ця перевага

мотивує вибір більш жадібного алгоритму, однак, якщо використовується ефективний алгоритм найближчого сусіда, на відміну від очевидного методу лінійного часу, тоді може мати сенс алгоритму бути менш жадібним.

Також є можливий варіант змусити CONNECT додати лише останню вершину в ітерації EXTEND до RRT, щоб зменшити кількість вузлів.

2.4 Алгоритм Extended-RRT

На відміну від алгоритму RRT-Connect, який в порівнянні зі стандартним алгоритмом RRT використовує двонаправлений пошук для прискорення планування дерева в ефективній структурі просторових даних, Extended-RRT використовує k-d-дерево для прискорення пошуку найближчого сусіда, але натомість не використовує двонаправлений пошук, оскільки він зменшує загальність специфікації цільового стану [27].

Додаткові можливі оптимізації включають більш загальний упереджений розподіл у формі кешу маршрутних точок. Якщо план шляху було знайдено під час попередньої ітерації, це, ймовірно, може допомогти під час повторного пошуку шляху пізніше. Простір конфігурації може змінитися, але зазвичай не дуже, тому дані стосовно попередніх побудованих шляхів може бути орієнтиром.

Кеш маршрутних точок реалізується шляхом збереження постійного розміру масиву станів, і щоразу, коли шлях знаходиться, усі стани точок шляху поміщаються в кеш із випадковою заміною. Це зберігає інформацію про те, де план побудованого шляху можна знову знайти в майбутньому.

З ERRT тепер існує дві ймовірності розподілу цільових станів (рис. 2.9):

– з імовірністю P (goal) q_{goal} вибирається як ціль;

– з імовірністю R (waypoint) вибирається випадкова точка шляху.

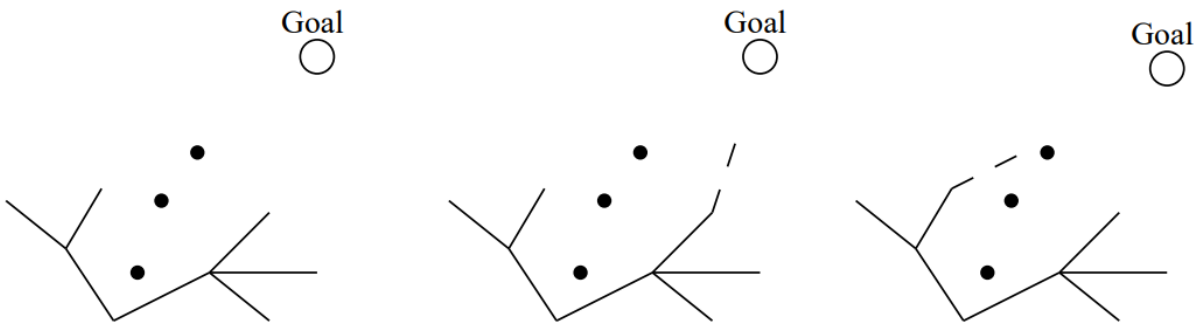

Рисунок 2.9 – Використання алгоритмом Extended-RRT кешу точок для перепланування маршруту

Простий планувальник RRT будує жадібне наближення до мінімального охоплюючого дерева і не дбає про довжину шляху від початкового стану. Показник відстані можна модифікувати, щоб включати не лише відстань від дерева до цільового стану, але й відстань від кореня дерева, помножену на деяке значення підсилення. Більше значення цього коефіцієнта підсилення призводить до коротших відстаней від кореня дерева до його листя, але також зменшує обсяг дослідження простору станів, зміщуючи його майже до початкового стану в «кущистому» дереві. При значенні в одиницю для коефіцієнта підсилення, алгоритм завжди буде поширювати листя дерева від кореневого вузла для будь-якої евклідової метрики в безперервній області, тоді як значення 0 еквівалентно вихідному алгоритму. Найкраще значення, здається, залежить від предметної області та навіть від екземпляру проблеми, і, здається, є гарним компромісом між пошуком коротшого плану та відсутністю його взагалі [28]. Однак при упередженому перепланування замість цього можна використовувати адаптивний механізм, який працює досить добре. Коли планувальник запускається, коефіцієнт підсилення встановлюється на 0. Потім під час послідовних перепланувань, якщо попередній запуск знайшов шлях, коефіцієнт підсилення збільшується, а в іншому випадку – зменшується. Цей адаптивний графік зміщення відображає ідею про те, що поганий маршрут кращий, ніж

відсутність маршруту спочатку і коли знайдений маршрут знаходиться в кеші, а пошук зміщений у бік маршрутних точок, алгоритм спонукає систему спробувати знайти більш оптимальній маршрут, допомагаючи покращити його під час послідовних прогонів.

2.5 Висновки до другого розділу

У розділі досліджено алгоритм RRT та його модифікації RRT^*, RRT-Connect, Extended-RRT, їх переваги та недоліки. Наведені приклади побудови дерева кожним з алгоритмів.

3 РОЗРОБКА ПРОГРАМИ ДЛЯ ПРОВЕДЕННЯ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

3.1 Розробка карти місцевості та підготовка до імітаційного моделювання

Для проведення імітаційного моделювання, де під імітаційним моделюванням розуміється створення комп'ютерної програми , для відтворення процесу функціонування складної системи у часі, створена об'ємна імітована карта місцевості функціонування робота маніпулятора, розміром 20 × 20 × 10 умовних одиниць. На цій карті, за допомогою створення на ній перешкод різної площі, будуть проводитися експерименти з пошуку шляху обраними алгоритмами [29]. Висота зазначених перешкод буде конфігуруватися згідно деякого відсотка від максимальної висоти карти і поступово збільшуватися, задля тестування обраних алгоритмів на предмет пошуку шляху саме у об'ємному середовищі. У результаті багатократних прогонів, бо властивістю обраних алгоритмів є саме непередбачуваний(рандомізований) пошук шляху при кожному новому запуску алгоритму, створена імітаційна модель покаже усереднене значення по обраним для тестування критеріям, а дослідник отримає дані щодо властивостей реальної системи.

Критеріями, обраними для тестування є:

– швидкість роботи алгоритму;

– довжина знайденого шляху;

– споживання пам'яті.

Для проведення моделювання, як було сказано вище, буде створено п'ять варіацій об'ємних карт з розмірами 20 × 20 × 10. Відрізнятися ці карти будуть кількістю перешкод різної площі, кількістю від 1 до 5 включно. Також, для тестування кожної карти, висоти перешкод також будуть змінюватися у відсотковому відношенні до максимальної висоти карти: 20 %, 40 %, 60 %, 80 %

відповідно. Роботу кожного алгоритму для кожної окремої варіації карти буде протестовано п’ять разів, показники будуть занесені до таблиць та буде виведено середнє значення для кожного з критеріїв.

3.2 Показники роботи алгоритмів в створеному середовищі

3.2.1 Алгоритм RRT

Показники роботи алгоритму на карті з однією перешкодою та її висотою у 40 % від максимальної висоти карти показані у таблиці 3.1.

Таблиця 3.1 – Показники роботи алгоритму RRT на карті з однією перешкодою

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	24	40,475	285,72
2	43	41,343	294,43
3	27	44,520	281,12
4	13	38,122	320,98
5	16	40,892	276,55
Середні значення	25	41,0704	291,76

Середні значення показників роботи алгоритму на карті з однією перешкодою для різних висот перешкоди наведені в таблиці 3.2.

Таблиця 3.2 – Середні показники роботи алгоритму RRT для різних висот перешкоди на карті з однією перешкодою

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	21	36,96	289,48
40 %	25	41,0704	291,76
60 %	26	49,71	301,93
80 %	25	50,40	312,56

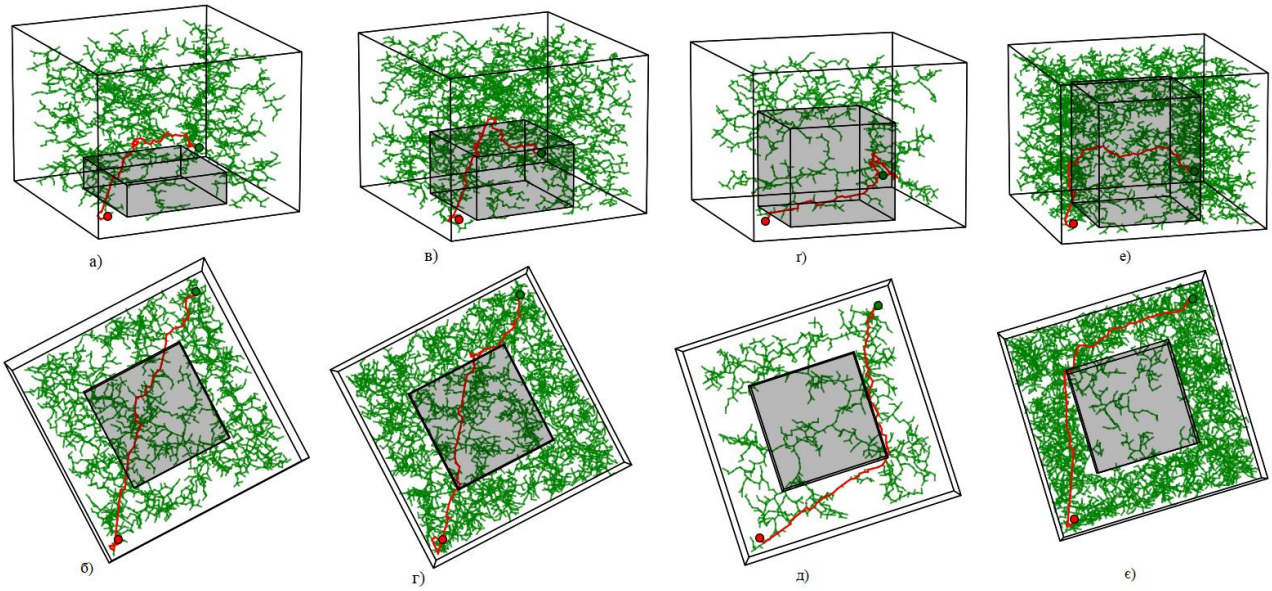

Рисунок 3.1 – Результати роботи алгоритму RRT при тестуванні на карті з однією перешкодою для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

З отриманих даних можна зробити висновок, що коли висота перешкод не перевищує 40 % від максимальної висоти карти, алгоритм RRT намагається дістатися цільової точки, обходячи перешкоду згори. Після 60 % видно, що даний алгоритм вже починає обходити перешкоду збоку. Обхід перешкод для карти з однією перешкодою зображено на рисунку 3.1.

Показники роботи алгоритму на карті з двома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.3.

Таблиця 3.3 – Показники роботи алгоритму RRT на карті з двома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	54	43,62	318,54
2	28	41,79	322,12
3	14	40,51	311,43
4	12	42,11	314,51
5	32	39,81	324,11
Середні значення	28	41,57	318,14

Середні значення показників роботи алгоритму на карті з двома перешкодами для різних висот перешкод наведені в таблиці 3.4.

Таблиця 3.4 – Середні показники роботи алгоритму RRT для різних висот перешкод на карті з двома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	24	36,57	315,13
40 %	28	41,57	318,14
60 %	27	50,484	332,53
80 %	28	50,618	338,4

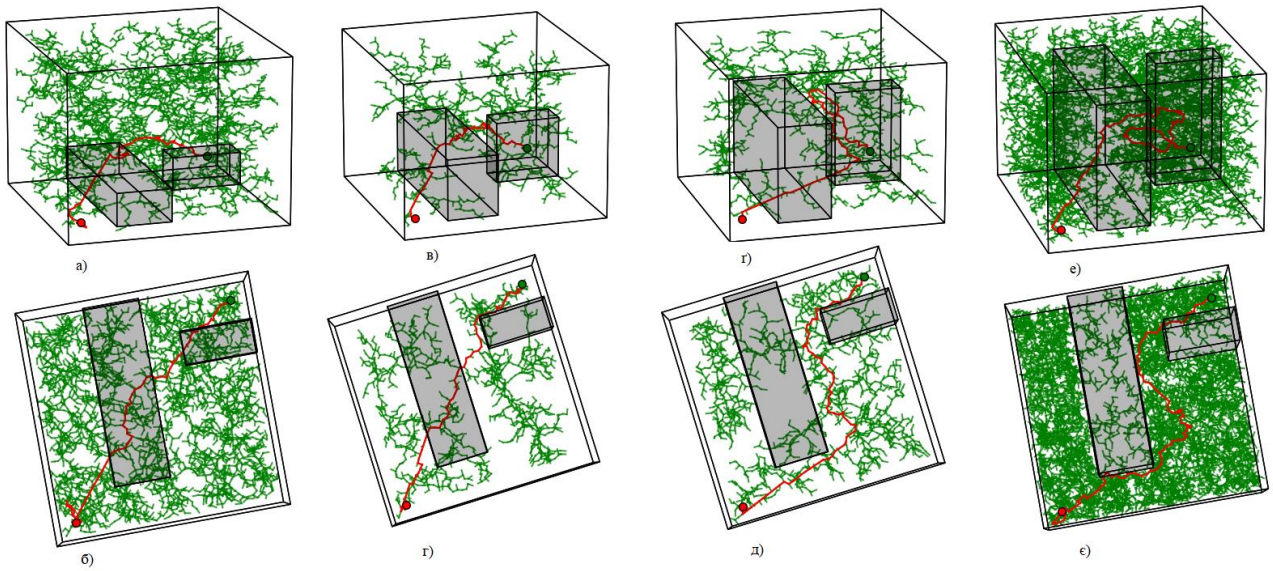

Рисунок 3.2 – Результати роботи алгоритму RRT при тестуванні на карті з двома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Результати експериментів стосовно обходу перешкод для карти з двома перешкодами збігаються з результатами експериментів для карти з однією перешкодою. Обхід перешкод для карти з двома перешкодами зображено на рисунку 3.2.

Показники роботи алгоритму на карті з трьома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.5.

Таблиця 3.5 – Показники роботи алгоритму RRT на карті з трьома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	8	44,90	312
2	56	42,43	322
3	34	41,12	317
4	12	40,86	315
5	18	41,45	323
Середні значення	25	42,152	317,8

Середні значення показників роботи алгоритму на карті з трьома перешкодами для різних висот перешкод наведені в таблиці 3.6.

Таблиця 3.6 – Середні показники роботи алгоритму RRT для різних висот перешкод на карті з трьома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	23	39,516	314,5
40 %	25	42,152	317,8
60 %	28	55,512	324,6
80 %	27	58,314	332,6

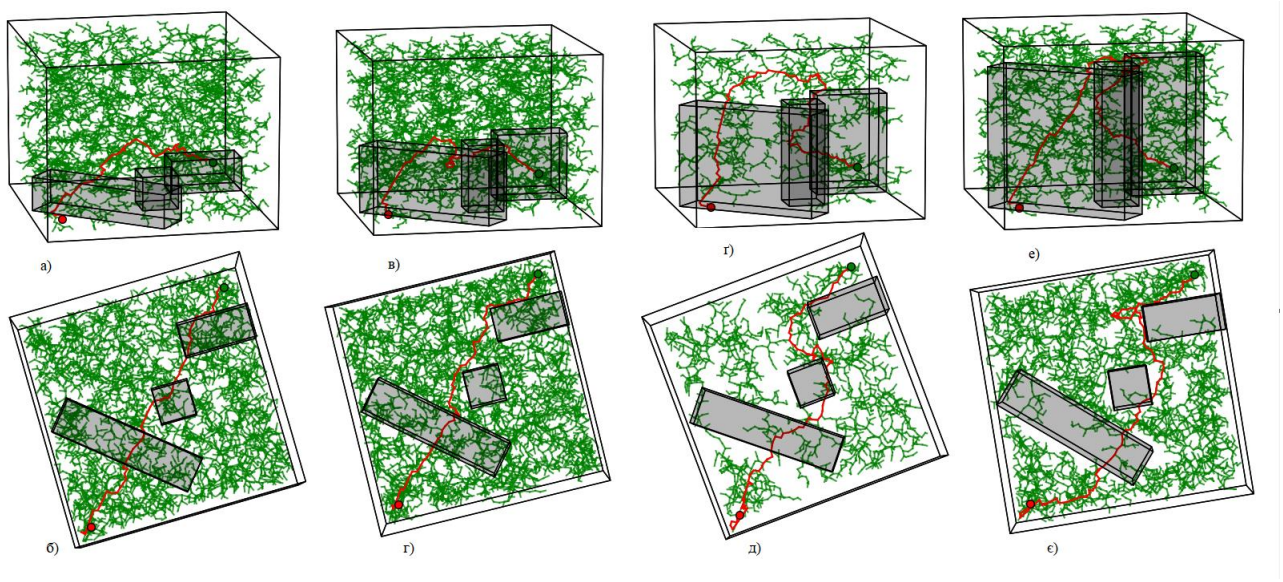

Рисунок 3.3 – Результати роботи алгоритму RRT при тестуванні на карті з трьома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Як видно з рисунку 3.3, на відміну від карт з однією та двома перешкодами, при знаходженні шляху на карті з трьома перешкодами, алгоритм RRT схиляється до обходження перешкод зверху, для будь якої з обраних висот перешкод.

Показники роботи алгоритму на карті з чотирма перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.7.

Таблиця 3.7 – Показники роботи алгоритму RRT на карті з чотирма перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	11	42,04	282
2	64	44,52	266
3	28	47,43	274
4	18	43,51	268
5	25	41,78	281
Середні значення	29	43,85	274,2

Середні значення показників роботи алгоритму на карті з чотирма перешкодами для різних висот перешкод наведені в таблиці 3.8.

Таблиця 3.8 – Середні показники роботи алгоритму RRT для різних висот перешкод на карті з чотирма перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	26	37,76	267,3
40 %	29	43,85	274,2
60 %	32	48,92	291,4
80 %	35	55,31	312,6

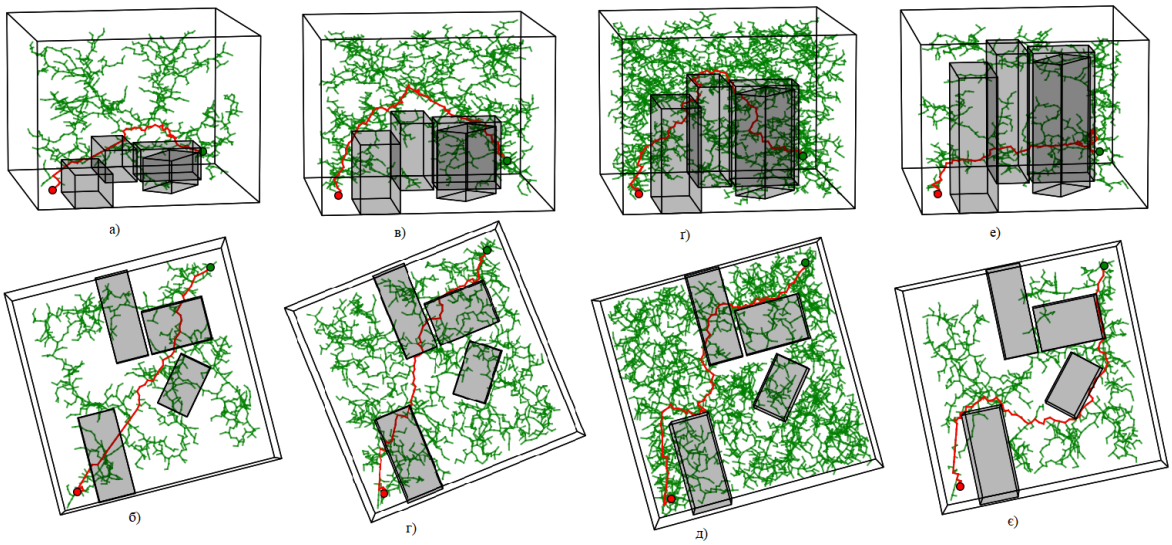

Рисунок 3.4 – Результати роботи алгоритму RRT при тестуванні на карті з чотирма перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Виходячи з рисунку 3.4 видно, що алгоритм RRT обходить перешкоди, висотою менше 60 % від максимальної висоти заданої карти, зверху, однак якщо висота перешкод більша за 80 % від максимальної, алгоритм починає обходити їх збоку.

3.2.2 Алгоритм RRT*

Показники роботи алгоритму на карті з однією перешкодою та її висотою у 40% від максимальної висоти карти показані у таблиці 3.9.

Таблиця 3.9 – Показники роботи алгоритму RRT^* на карті з однією перешкодою

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	41	38,69	315
2	38	39,12	317
3	40	38,24	311
4	39	40,11	298
5	41	37,65	314
Середні значення	40	38,76	311

Середні значення показників роботи алгоритму на карті з однією перешкодою для різної висоти перешкоди наведені в таблиці 3.10.

Таблиця 3.10 – Середні показники роботи алгоритму RRT^* для різних висот перешкоди на карті з однією перешкодою

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	38	33,77	326
40 %	40	38,76	311
60 %	40	45,81	315
80 %	41	47,92	315

Виходячи з даних наведених у таблиці 3.9 можна зробити висновок, що для карти з однією перешкодою, показники швидкості роботи алгоритму RRT^* більш стабільні за показники стандартного RRT.

Стосовно даних наведених у таблиці 3.10 та на рисунку 3.5, можна побачити, що алгоритм RRT^* в цілому знаходить шлях коротший, ніж алгоритм RRT і починає обходити перешкоду збоку лише після того, коли перешкода займає 80 % від максимальної висоти карти, в той час як стандартний RRT починає обходити її збоку вже після того, як висота перешкоди дорівнює 60 % від максимальної висоти карти, що, зокрема, також призводить до скорочення знайденого маршруту для даної конфігурації карти.

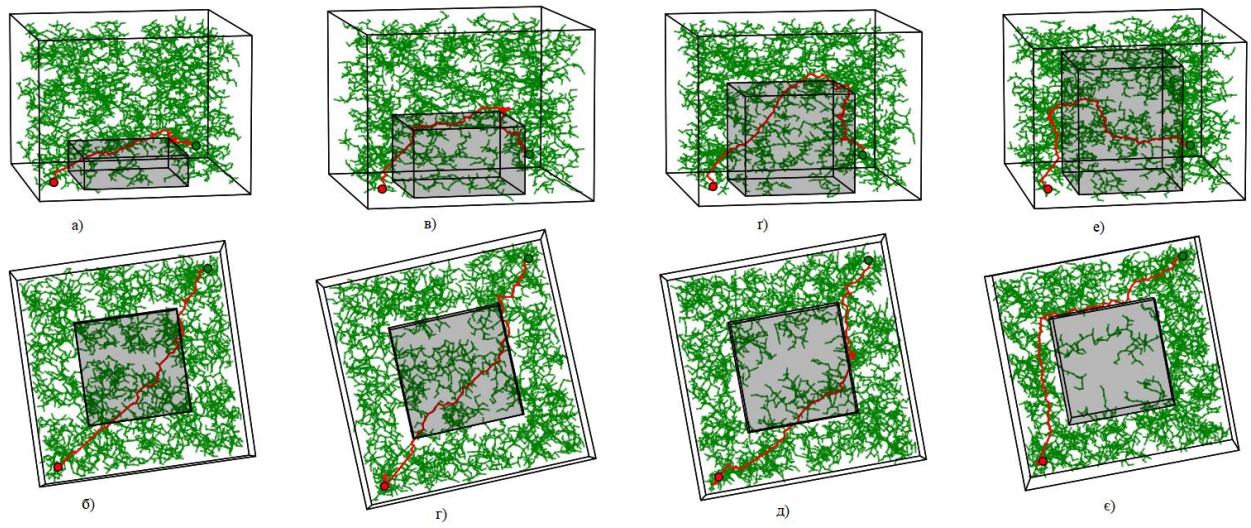

Рисунок 3.5 – Результати роботи алгоритму RRT^* при тестуванні на карті з однією перешкодою для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з двома перешкодами та їх висотою у 40% від максимальної висоти карти показані у таблиці 3.11.

Таблиця 3.11 – Показники роботи алгоритму RRT^* на карті з двома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	2	3	4
1	44	39,32	316
2	41	38,13	318
3	44	39,28	314

Продовження таблиці 3.11

1	2	3	4
4	39	41,26	322
5	39	39,41	318
Середні значення	41	39,48	317

Середні значення показників роботи алгоритму на карті з двома перешкодами для різних висот перешкод наведені в таблиці 3.12.

Таблиця 3.12 – Середні показники роботи алгоритму RRT^* для різних висот перешкод на карті з двома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	41	35,73	316
40 %	41	39,48	317
60 %	42	47,32	320
80 %	42	47,36	323

З рисунку 3.6 видно шо алгоритм RRT^* подібно стандартному алгоритму RRT починає обходити перешкоди збоку, після того як висота перешкод сягає 60 % від максимальної висоти карти. При цьому з таблиці 3.12 видно, що алгоритм RRT^* шукає коротший маршрут, але продовжує працювати довше за стандартний RRT.

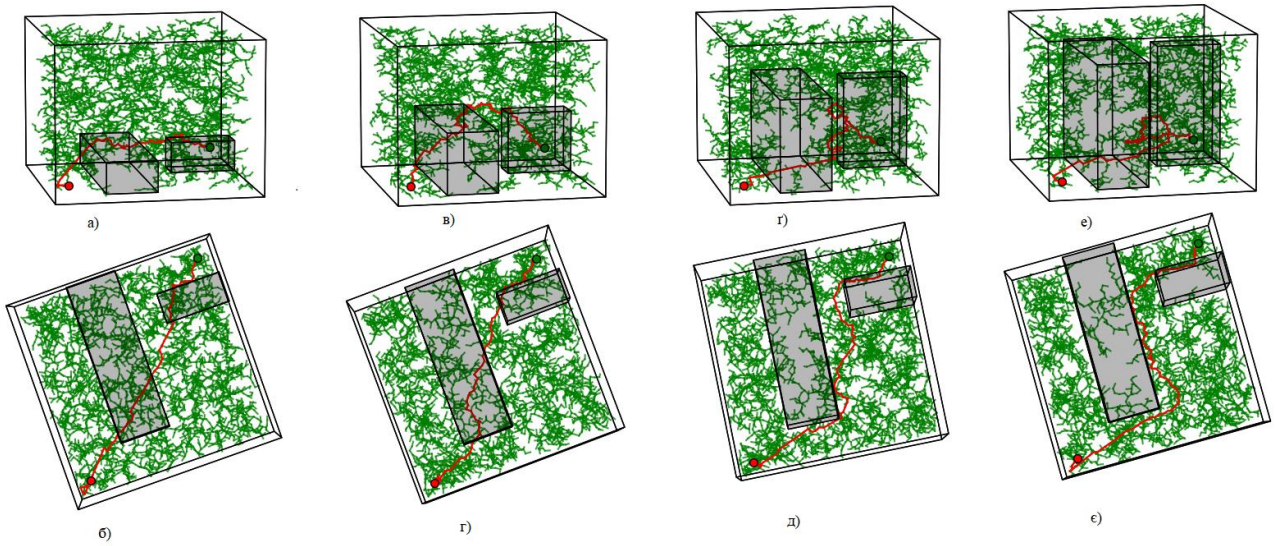

Рисунок 3.6 – Результати роботи алгоритму RRT^* при тестуванні на карті з двома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з трьома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.13.

Таблиця 3.13 – Показники роботи алгоритму RRT^* на карті з трьома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	2	3	4
1	43	38,92	315
2	43	40,45	318
3	44	41,12	320

Продовження таблиці 3.13

1	2	3	4
4	42	38,22	314
5	43	40,94	319
Середні значення	43	39,93	317

Середні значення показників роботи алгоритму на карті з трьома перешкодами для різних висот перешкод наведені в таблиці 3.14.

Таблиця 3.14 – Середні показники роботи алгоритму RRT^* для різних висот перешкод на карті з трьома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (сек.)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (МБ)
20 %	42	35,81	329
40 %	43	39,93	317
60 %	42	50,95	321
80 %	42	54,15	318

На рисунку 3.6 можна побачити, що алгоритм RRT^*, як і стандартний алгоритм RRT намагається обходити перешкоди згори, якщо це робить шлях коротшим, але якщо це не потрібно, починає обходити їх збоку, на відміну від стандартного алгоритму. Також за результатами експерименту (таб. 3.14) видно, що алгоритм RRT^* працює швидше за стандартний алгоритм RRT і на карті з трьома перешкодами.

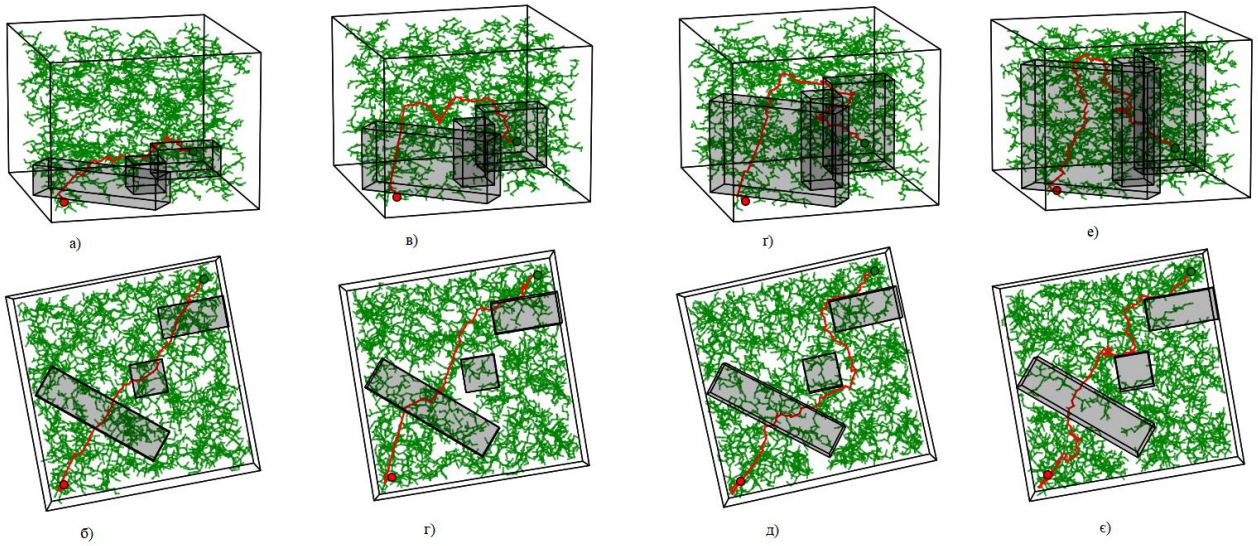

Рисунок 3.6 – Результати роботи алгоритму RRT^* при тестуванні на карті з трьома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з чотирма перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.15.

Таблиця 3.15 – Показники роботи алгоритму RRT^* на карті з чотирма перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	2	3	4
1	45	42,86	322
2	42	39,64	320
3	41	39,81	320

Продовження таблиці 3.15

1	2	3	4
4	42	40,55	321
5	44	41,08	321
Середні значення	43	40,78	320

Середні значення показників роботи алгоритму на карті з чотирма перешкодами для різних висот перешкод наведені в таблиці 3.16.

Таблиця 3.16 – Середні показники роботи алгоритму RRT^* для різних висот перешкод на карті з чотирма перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	41	34,52	318
40 %	42	40,78	320
60 %	43	46,56	314
80 %	41	52,27	313

З даних, отриманих в ході експерименту, які можна побачити у таблиці 3.16 та на рисунку 3.8, можна зробити висновок, що алгоритм RRT^* намагається знайти найкоротший шлях вдаючись до обходу перешкод зверху, коли це потрібно, і нехтуючи цим, якщо обхід перешкод збоку дасть кращий результат.

Також потрібно зазначити, що і в цьому експерименті, алгоритм RRT^* знаходить шлях коротший за стандартний алгоритм RRT, хоч це і займає більше

часу ніж у останнього, але треба зазначити, що на відміну від, наприклад, карти з однією перешкодою, різниця в часі пошуку маршруту вже не так помітна.

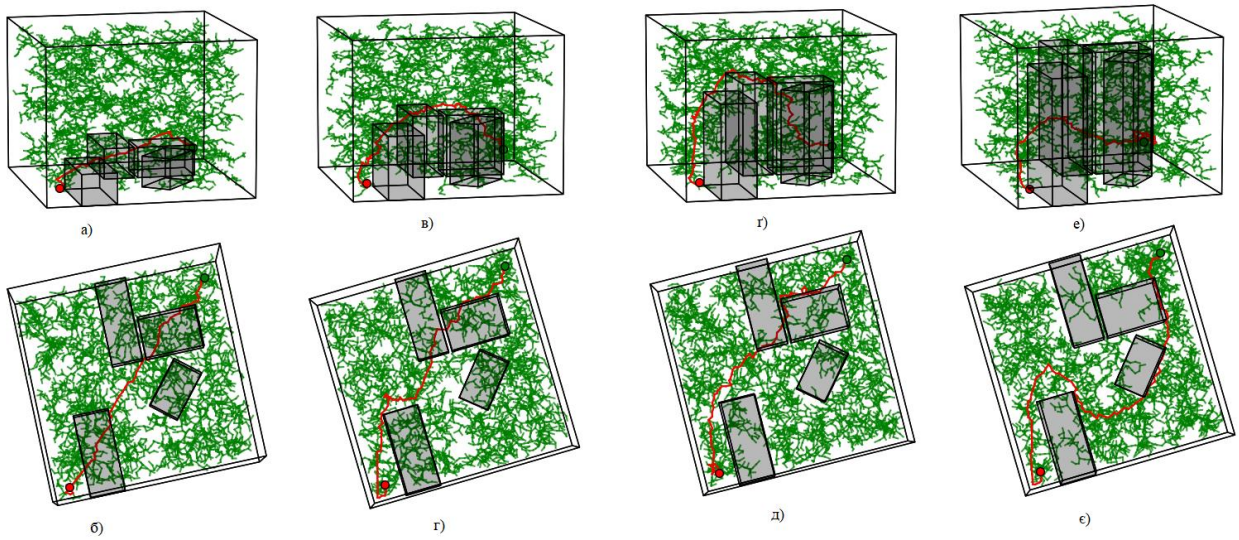

Рисунок 3.8 – Результати роботи алгоритму RRT^* при тестуванні на карті з чотирма перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

3.2.3 Алгоритм RRT-Connect

Показники роботи алгоритму на карті з однією перешкодою та її висотою у 40 % від максимальної висоти карти показані у таблиці 3.17.

Середні значення показників роботи алгоритму на карті з однією перешкодою для різної висоти перешкоди наведені в таблиці 3.18.

Таблиця 3.17 – Показники роботи алгоритму RRT-Connect на карті з однією перешкодою

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,2	48,14	258
2	1,3	45,28	260
3	0,9	47,11	261
4	1,2	44,52	256
5	1	45,22	260
Середні значення	1,12	46,05	259

Таблиця 3.18 – Середні показники роботи алгоритму RRT-Connect для різних висот перешкоди на карті з однією перешкодою

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	0,96	39,95	258
40 %	1,12	46,05	259
60 %	2,13	56,42	268
80 %	2,19	57,11	274

З даних, отриманих в ході експерименту (таб. 3.18), видно, що на відміну від стандартного алгоритму RRT, RRT-Connect працює значно швидше та споживає менше пам’яті, але, як можна побачити на рисунку 3.9, алгоритм RRT-Connect знаходить довший шлях, аніж стандартний RRT.

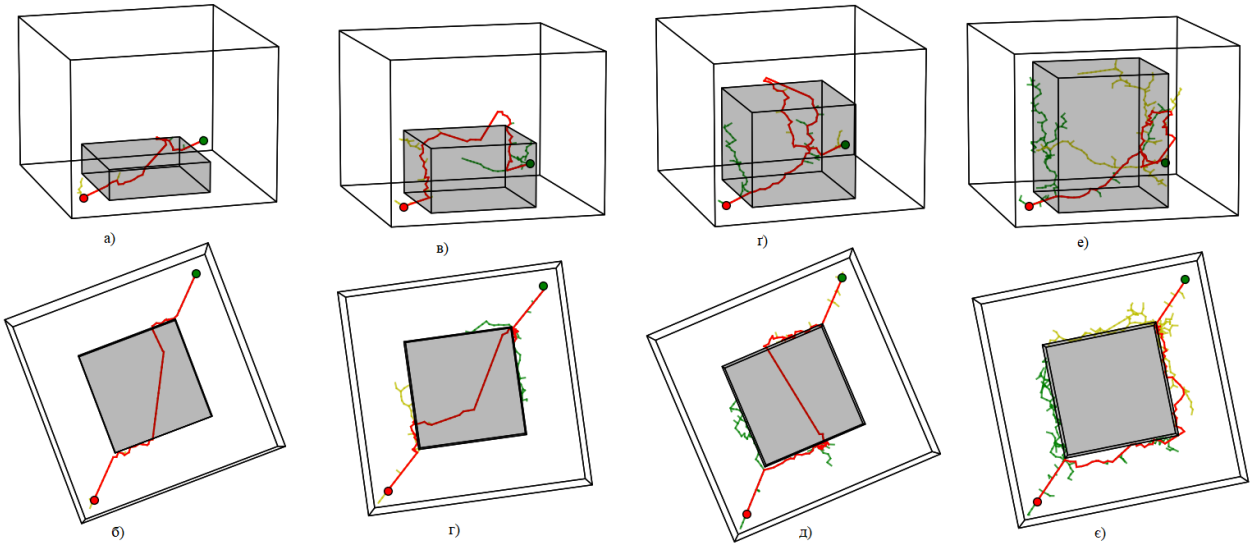

Рисунок 3.9 – Результати роботи алгоритму RRT-Connect при тестуванні на карті з однією перешкодою для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з двома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.19.

Таблиця 3.19 – Показники роботи алгоритму RRT-Connect на карті з двома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	2	3	4
1	1,2	48,12	264
2	1,2	47,28	266

Продовження таблиці 3.19

1	2	3	4
3	1,3	45,72	262
4	1,3	44,76	258
5	1,3	46,51	266
Середні значення	1,26	46,48	263

Середні значення показників роботи алгоритму на карті з двома перешкодами для різних висот перешкод наведені в таблиці 3.20.

Таблиця 3.20 – Середні показники роботи алгоритму RRT-Connect для різних висот перешкод на карті з двома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	1,2	40,11	258
40 %	1,26	46,48	263
60 %	2,1	50,46	264
80 %	2,5	56,70	274

З рисунку 3.10 видно, що алгоритм RRT-Connect починає обходити перешкоди збоку, тільки коли висота перешкод сягає 80 % від максимальної висоти карти, при висоті до 60 % від максимальної висоти карти, намагається обходити їх згори, що впливає довжину маршруту в порівнянні зі стандартним RRT. Також за результатами експерименту видно, що алгоритм RRT-Connect

продовжує працювати значно швидше за попередні алгоритми та споживає менше пам’яті, але в середньому знаходить маршрут довший, ніж вони.

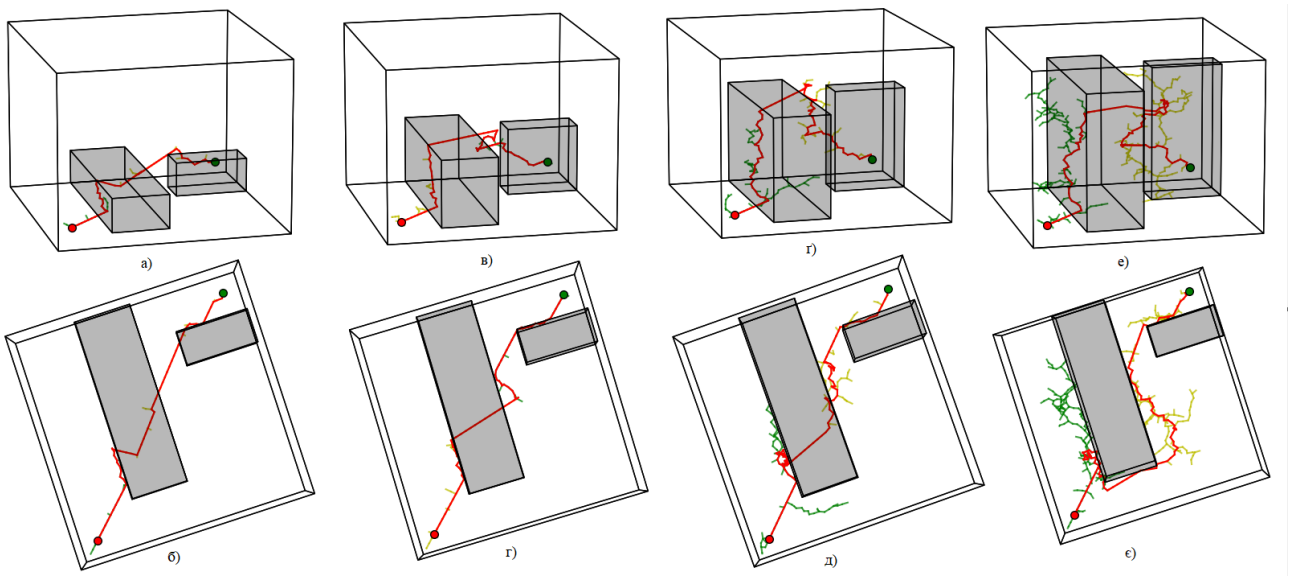

Рисунок 3.10 – Результати роботи алгоритму RRT-Connect при тестуванні на карті з двома перешкодами для різних висот

На рис. 3.10 є такі позначення.

а – для висоти перешкоди у 20 %.

б – вид зверху.

в – для висоти перешкоди у 40 %.

г – вид зверху.

ґ – для висоти перешкоди у 60 %.

д – вид зверху.

е – для висоти перешкоди у 80 %.

є – вид зверху.

Показники роботи алгоритму на карті з трьома перешкодами та їх висотою у 40% від максимальної висоти карти показані у таблиці 3.21.

Середні значення показників роботи алгоритму на карті з трьома перешкодами для різних висот перешкод наведені в таблиці 3.22.

Таблиця 3.21 – Показники роботи алгоритму RRT-Connect на карті з трьома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,07	47,62	268
2	1,08	48,13	267
3	1,07	45,27	266
4	1,07	45,11	267
5	1,08	46,52	266
Середні значення	1,07	46,53	267

Таблиця 3.22 – Середні показники роботи алгоритму RRT-Connect для різних висот перешкод на карті з трьома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	1,04	40,88	265
40 %	1,07	46,53	267
60 %	2,23	58,11	271
80 %	6,25	62,72	279

За результатами даного експерименту (таб. 3.22) можна сказати, що для карт на яких є три перешкоди, алгоритм RRT-Connect, при відносно малій висоті перешкод, знаходить маршрут який не сильно відрізняється від маршруту, знайденому стандартним RRT, але робить це значно швидше. Також варто

зазначити, що при висоті перешкод у 80 % від максимальної висоти, алгоритм RRT-Connect значно сповільнюється та сильно зростає довжина знайденого маршруту (рис 3.11).

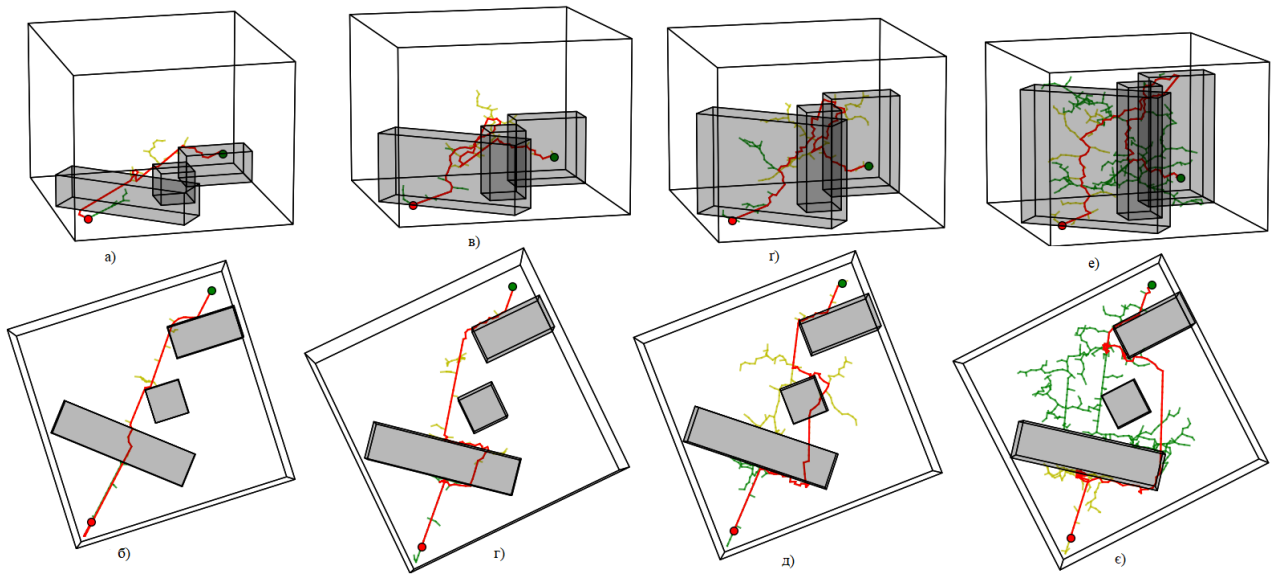

Рисунок 3.11 – Результати роботи алгоритму RRT-Connect при тестуванні на карті з двома перешкодами для різних висот

На рис. 3.11 є такі позначення.

а – для висоти перешкоди у 20 %.

б – вид зверху.

в – для висоти перешкоди у 40 %.

г – вид зверху.

ґ – для висоти перешкоди у 60 %.

д – вид зверху.

е – для висоти перешкоди у 80 %.

є – вид зверху.

Показники роботи алгоритму на карті з чотирма перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.23.

Таблиця 3.23 – Показники роботи алгоритму RRT-Connect на карті з чотирма перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,4	47,71	264
2	1,3	46,56	265
3	1,4	46,56	265
4	1,4	44,82	264
5	1,4	46,86	264
Середні значення	1,4	46,50	264

Середні значення показників роботи алгоритму на карті з трьома перешкодами для різних висот перешкод наведені в таблиці 3.24.

Таблиця 3.24 – Середні показники роботи алгоритму RRT-Connect для різних висот перешкод на карті з трьома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	0,9	40,32	261
40 %	1,4	46,50	264
60 %	1,9	53,26	271
80 %	6,5	60,43	281

З експерименту видно, що як і для карти з трьома перешкодами, для карти із чотирма перешкодами, при висоті перешкод у 80 % від максимальної висоти карти, алгоритм RRT-Connect починає сильно сповільнюватись і знаходить значно довший шлях у порівнянні з попередніми алгоритмами, відносно інших висот перешкод для даної карти.

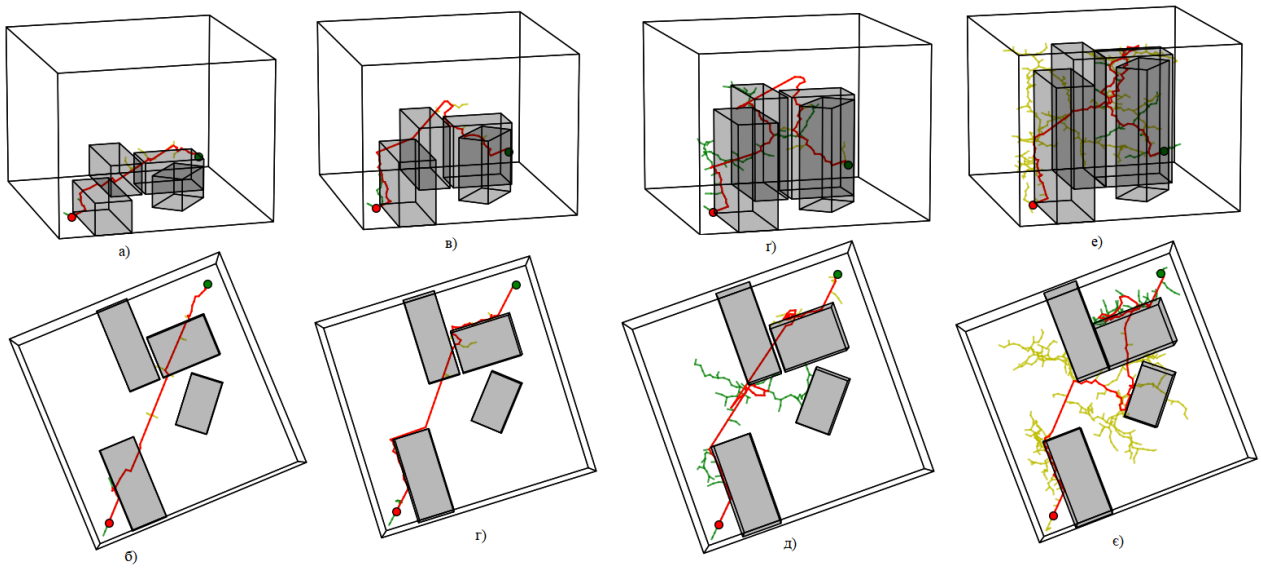

Рисунок 3.12 – Результати роботи алгоритму RRT-Connect при тестуванні на карті з чотирма перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

3.2.4 Алгоритм Extended-RRT

Показники роботи алгоритму на карті з однією перешкодою та її висотою у 40 % від максимальної висоти карти показані у таблиці 3.25.

Таблиця 3.25 – Показники роботи алгоритму Extended-RRT на карті з однією перешкодою

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,4	46,41	434
2	1,4	43,43	434
3	1,35	44,21	435
4	1,4	44,37	435
5	1,4	45,03	434
Середні значення	1,39	44,69	434

Середні значення показників роботи алгоритму на карті з однією перешкодою для різної висоти перешкоди наведені в таблиці 3.26.

Таблиця 3.26 – Середні показники роботи алгоритму Extended-RRT для різних висот перешкоди на карті з однією перешкодою

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	1,35	39,50	430
40 %	1,39	44,69	434
60 %	1,4	52,41	434
80 %	1,4	49,60	434

З експерименту видно, що для карти з однією перешкодою, для висоти перешкоди не вище ніж 40 % від максимальної висоти карти, алгоритм Extended-RRT програє стандартному RRT та RRT^* , але виграє у RRT-Connect в довжині побудованого маршруту, а при висоті вище ніж 80 % від максимальної висоти карти, починає вигравати усіх трьох. Що стосується швидкості роботи алгоритму, Extended-RRT програє тільки алгоритму RRT-Connect, а у розмірі споживаної пам’яті програє усім.

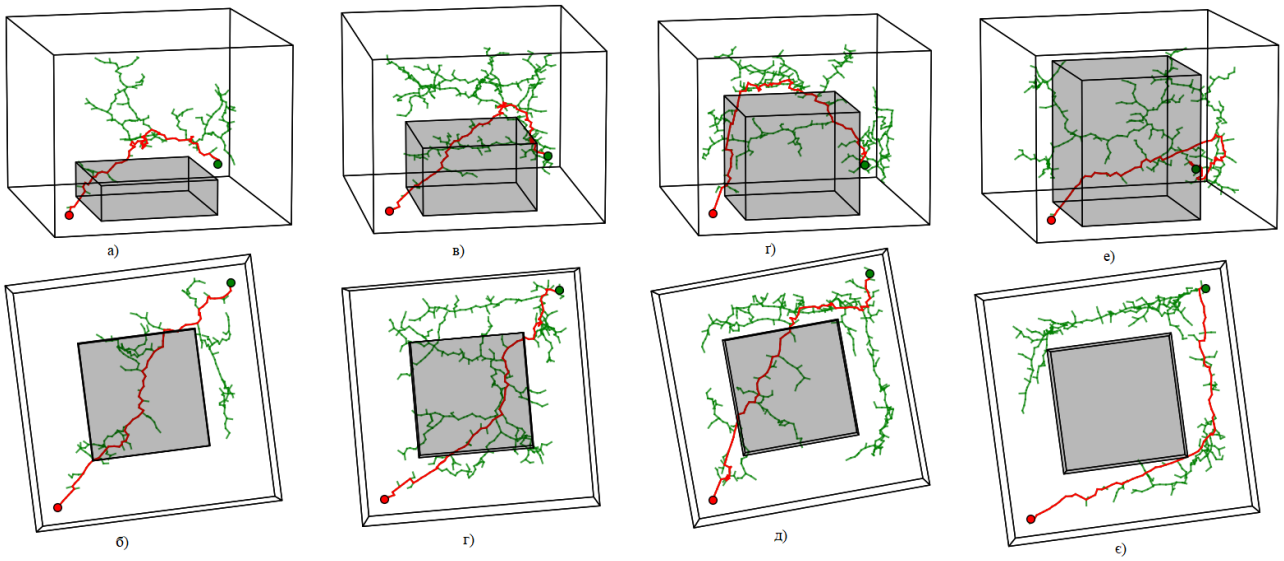

Рисунок 3.13 – Результати роботи алгоритму Extended-RRT при тестуванні на карті з чотирма перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з двома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.27.

Таблиця 3.27 – Показники роботи алгоритму Extended-RRT на карті з двома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,53	46,82	430
2	1,55	46,41	432
3	1,53	42,28	432
4	1,54	43,74	435
5	1,54	44,03	432
Середні значення	1,54	44,66	432

Середні значення показників роботи алгоритму на карті з двома перешкодами для різних висот перешкод наведені в таблиці 3.28.

Таблиця 3.28 – Середні показники роботи алгоритму Extended-RRT для різних висот перешкод на карті з двома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	1,53	39,82	431
40 %	1,54	44,66	432
60 %	1,54	46,85	432
80 %	1,54	47,22	435

З результатів експерименту (таб. 3.28) видно, що за висоти перешкод не вище 40 % від максимальної висоти карти, довжина знайденого шляху алгоритмом Extended-RRT програє довжині шляху, знайденій алгоритмами RRT та RRT^* але виграє у довжини, знайденої алгоритмом RRT-Connect. При висоті віще 60 %, Extended-RRT виграє у всіх попередніх алгоритмів.

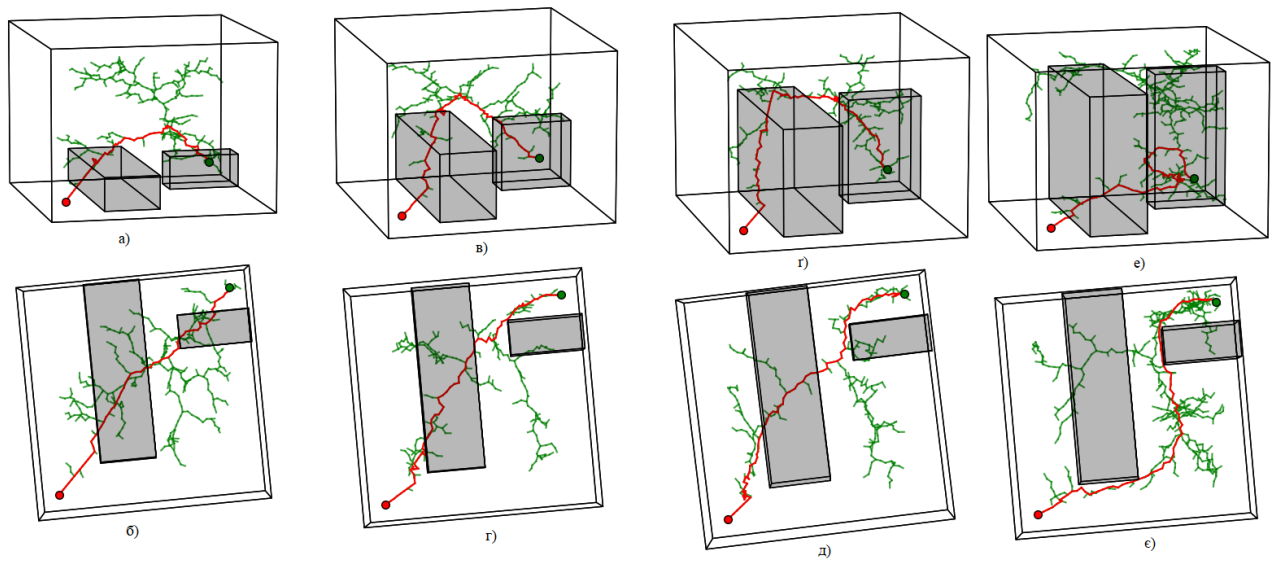

Рисунок 3.14 – Результати роботи алгоритму Extended-RRT при тестуванні на карті з двома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з трьома перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.29.

Таблиця 3.29 – Показники роботи алгоритму Extended-RRT на карті з трьома перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	1,7	44,63	432
2	1,8	44,82	432
3	1,7	41,93	431
4	1,7	43,33	434
5	1,7	42,52	432
Середні значення	1,7	43,45	432

Середні значення показників роботи алгоритму на карті з трьома перешкодами для різних висот перешкод наведені в таблиці 3.28.

Таблиця 3.30 – Середні показники роботи алгоритму Extended-RRT для різних висот перешкод на карті з трьома перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	1,6	39,81	431
40 %	1,7	43,45	432
60 %	1,8	46,86	434
80 %	3,6	50,36	435

На рисунку 3.15 видно, що довжина маршруту, побудованого алгоритмом Extended-RRT, для висоти перешкод не вище 40 % від максимальної висоти карти, більша за довжину маршруту, побудованого алгоритмами RRT та RRT^*, але менша за довжину маршруту, побудованого алгоритмом RRT-Connect. При висоті вище 60 % від максимальної висоти карти, Extended-RRT виграє у всіх трьох у довжині побудованого маршруту.

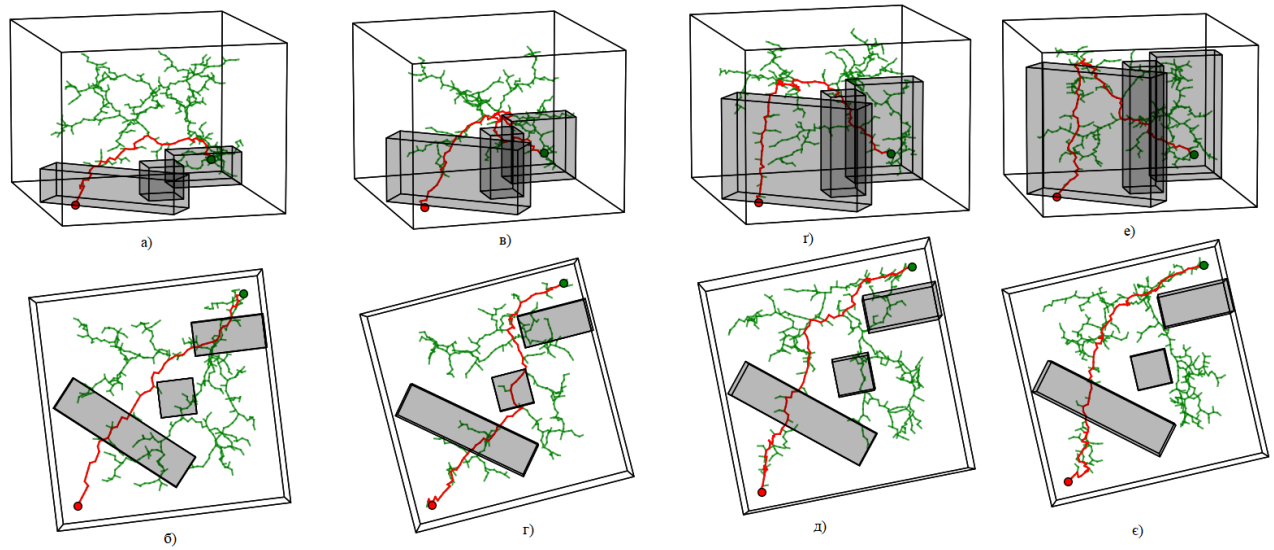

Рисунок 3.15 – Результати роботи алгоритму Extended-RRT при тестуванні на карті з трьома перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

Показники роботи алгоритму на карті з чотирма перешкодами та їх висотою у 40 % від максимальної висоти карти показані у таблиці 3.31.

Таблиця 3.31 – Показники роботи алгоритму Extended-RRT на карті з чотирма перешкодами

№ Ітерації	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
1	2,14	45,50	437
2	2,13	43,33	437
3	2,14	44,61	438
4	2,15	45,52	438
5	2,15	43,62	437
Середні значення	2,14	44,51	437

Середні значення показників роботи алгоритму на карті з чотирма перешкодами для різних висот перешкод наведені в таблиці 3.28.

Таблиця 3.32 – Середні показники роботи алгоритму Extended-RRT для різних висот перешкод на карті з чотирма перешкодами

Висота перешкоди від максимальної висоти карти (%)	Швидкість роботи алгоритму (с)	Довжина знайденого шляху (ум. о.)	Споживана пам’ять (Мб)
20 %	2,11	38,87	435
40 %	2,14	44,51	437
60 %	2,8	45,57	440
80 %	4,6	50,83	443

Як можна побачити за результатами, наведеними в таблиці 3.32, для карти з чотирма перешкодами, при висоті перешкод не вище 40 %, алгоритм Extended-RRT будує довший шлях, ніж алгоритми RRT та RRT^*, але коротший ніж шлях, побудований алгоритмом RRT-Connect. При висоті перешкод вище 60 % від максимальної висоти карти, Extended-RRT будує найкоротший шлях.

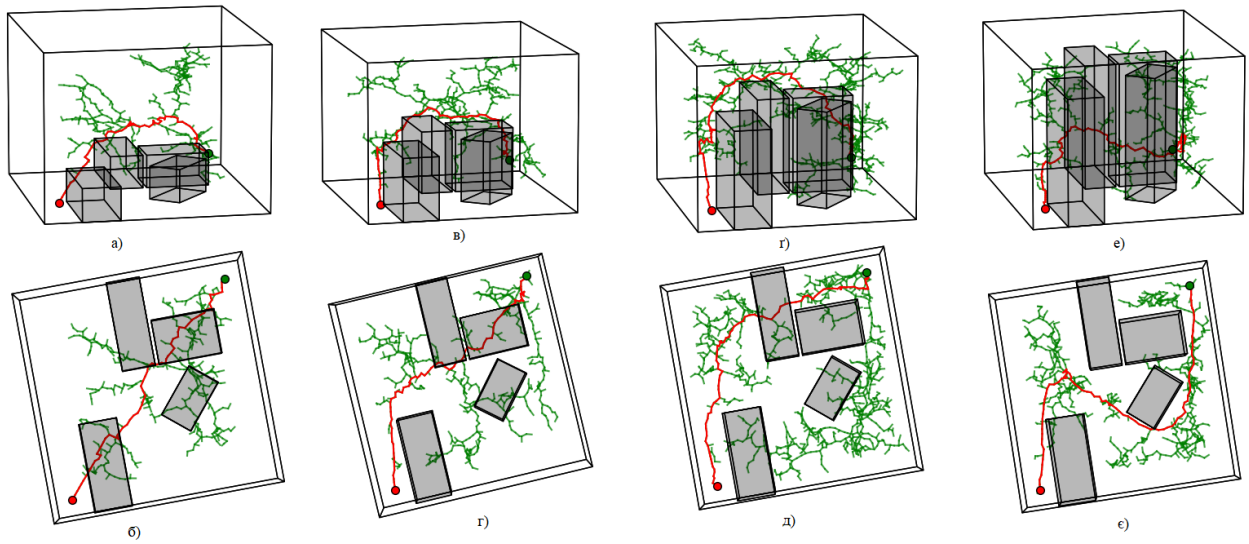

Рисунок 3.16 – Результати роботи алгоритму Extended-RRT при тестуванні на карті з чотирма перешкодами для різних висот (а – для висоти перешкоди у 20 %, б – вид зверху, в – для висоти перешкоди у 40 %, г – вид зверху, ґ – для висоти перешкоди у 60 %, д – вид зверху, е – для висоти перешкоди у 80 %, є – вид зверху)

3.3 Результати експериментів з алгоритмами пошуку шляху в 3D просторі для маніпуляторів

Порівняння показників, отриманих в ході експериментів з алгоритмами для пошуку шляху о об’ємному просторі для маніпуляторів, приведені у таблицях 3.33 – 3.36. Отримані числові результати підготовані для наступного використання при створенні системи підтримки прийняття рішень, тобто представленні у вигляді відношень одне до одного, де одиниця є найліпшим

результатом для кожного з критеріїв, а решта числових значень є показником у скільки разів той чи інший показник, для кожного з критеріїв, гірший за найліпший.

Таблиця 3.33 – Отримані в ході експерименту числові показники порівняння алгоритмів пошуку шляху в об’ємному просторі для маніпуляторі за кожним з критеріїв, для карт з однією та двома перешкодами

Висота	Назва	Одна перешкода			Дві перешкоди		
		Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)	Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)
Перешкоди висотою у 20 % від максимальної висоти карти	RRT	21,87	1,09	1,12	20	1,02	1,22
	RRT^*	39,58	1	1,26	34,16	1	1,22
	RRT-Connect	1	1,18	1	1	1,12	1
	Extended-RRT	1,40	1,17	1,66	1,27	1,11	1,67
Перешкоди висотою у 40 % від максимальної висоти карти	RRT	22,32	1,06	1,13	22,22	1,05	1,21
	RRT^*	35,71	1	1,20	32,54	1	1,2
	RRT-Connect	1	1,19	1	1	1,18	1
	Extended-RRT	1,24	1,15	1,67	1,22	1,13	1,64

Порівняння показників, отриманих в ході експериментів з алгоритмами для пошуку шляху о об’ємному просторі для маніпуляторів, приведені у таблицях 3.34

Таблиця 3.34 – Отримані в ході експерименту числові показники порівняння алгоритмів пошуку шляху в об’ємному просторі для маніпуляторі за кожним з критеріїв, для карт з однією та двома перешкодами

Висота	Назва	Одна перешкода			Дві перешкоди		
		Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)	Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)
Перешкоди висотою у 60 % від максимальної висоти карти	RRT	18,57	1,08	1,13	17,53	1,08	1,26
	RRT^*	28,57	1	1,17	27,27	1,01	1,21
	RRT-Connect	1,52	1,23	1	1,36	1,12	1
	Extended-RRT	1	1,08	1,62	1	1	1,64
Перешкоди висотою у 80 % від максимальної висоти карти	RRT	17,86	1,09	1,14	18,18	1,07	1,23
	RRT^*	29,28	1,04	1,15	27,27	1,003	1,18
	RRT-Connect	1,56	1,24	1	1,62	1,2	1
	Extended-RRT	1	1	1,58	1	1	1,59

Порівняння показників, отриманих в ході експериментів з алгоритмами для пошуку шляху о об’ємному просторі для маніпуляторів, приведені у таблиці 3.35.

Таблиця 3.35 – Отримані в ході експерименту числові показники порівняння алгоритмів пошуку шляху в об’ємному просторі для маніпуляторі за кожним з критеріїв, для карт з трьома та чотирма перешкодами

Висота	Назва	Три перешкоди			Чотири перешкоди		
		Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)	Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)
Перешкоди висотою у 20 % від максимальної висоти карти	RRT	22,11	1,10	1,19	28,89	1,09	1,02
	RRT^*	40,38	1	1,24	45,55	1	1,21
	RRT-Connect	1	1,14	1	1	1,17	1
	Extended-RRT	1,54	1,11	1,63	2,34	1,13	1,66
Перешкоди висотою у 40 % від максимальної висоти карти	RRT	23,36	1,05	1,19	20,71	1,07	1,03
	RRT^*	40,19	1	1,19	30	1	1,21
	RRT-Connect	1	1,16	1	1	1,14	1
	Extended-RRT	1,59	1,09	1,62	1,53	1,09	1,65

Порівняння показників, отриманих в ході експериментів з алгоритмами для пошуку шляху о об’ємному просторі для маніпуляторів, приведені у таблиці 3.36.

Таблиця 3.36 – Отримані в ході експерименту числові показники порівняння алгоритмів пошуку шляху в об’ємному просторі для маніпуляторі за кожним з критеріїв, для карт з трьома та чотирма перешкодами

Висота	Назва	Три перешкоди			Чотири перешкоди		
		Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)	Ш. р. (с)	Д. ш. (у.о.)	С. п. (Мб)
Перешкоди висотою у 60 % від максимальної висоти карти	RRT	15,55	1,18	1,19	16,84	1,07	1,07
	RRT^*	23,33	1,09	1,18	22,63	1,02	1,16
	RRT-Connect	1,23	1,24	1	1	1,17	1
	Extended-RRT	1	1	1,60	1,47	1	1,62
Перешкоди висотою у 80 % від максимальної висоти карти	RRT	7,5	1,15	1,19	7,61	1,09	1,11
	RRT^*	11,66	1,07	1,14	8,91	1,03	1,11
	RRT-Connect	1,74	1,24	1	1,41	1,19	1
	Extended-RRT	1	1	1,56	1	1	1,58

На рисунку 3.17 наведено порівняння відношень усереднених показників роботи алгоритмів пошуку шляху для маніпулятора за обраними критеріями, для висот перешкод у 20 % від максимальної висоти карти.

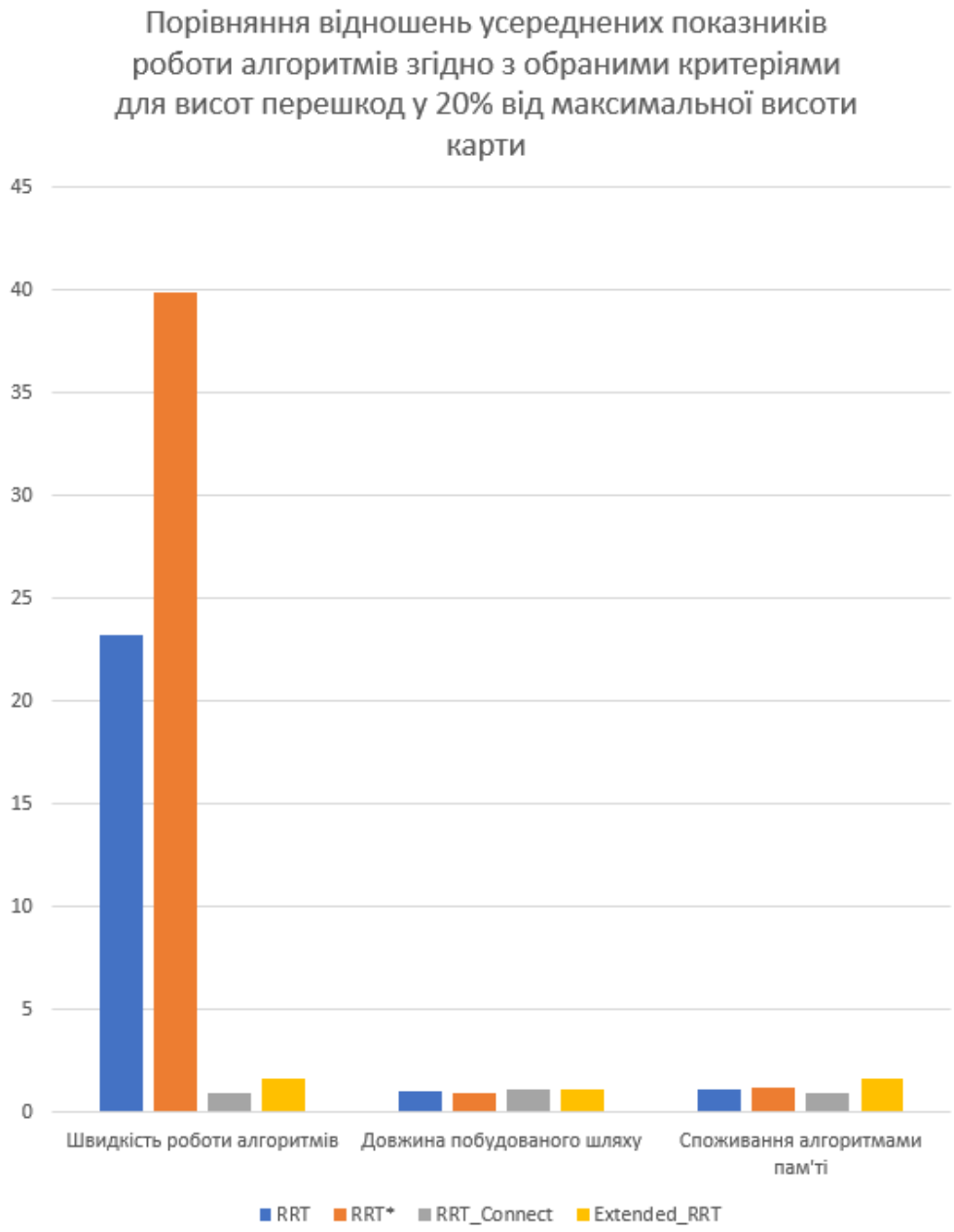

Рисунок 3.17 – Порівняння відношень усереднених показників роботи алгоритмів згідно з обраними критеріями для висот перешкод у 20 % від максимальної висоти карти

На рисунку 3.18 наведено порівняння відношень усереднених показників роботи алгоритмів пошуку шляху для маніпулятора за обраними критеріями, для висот перешкод у 40 % від максимальної висоти карти.

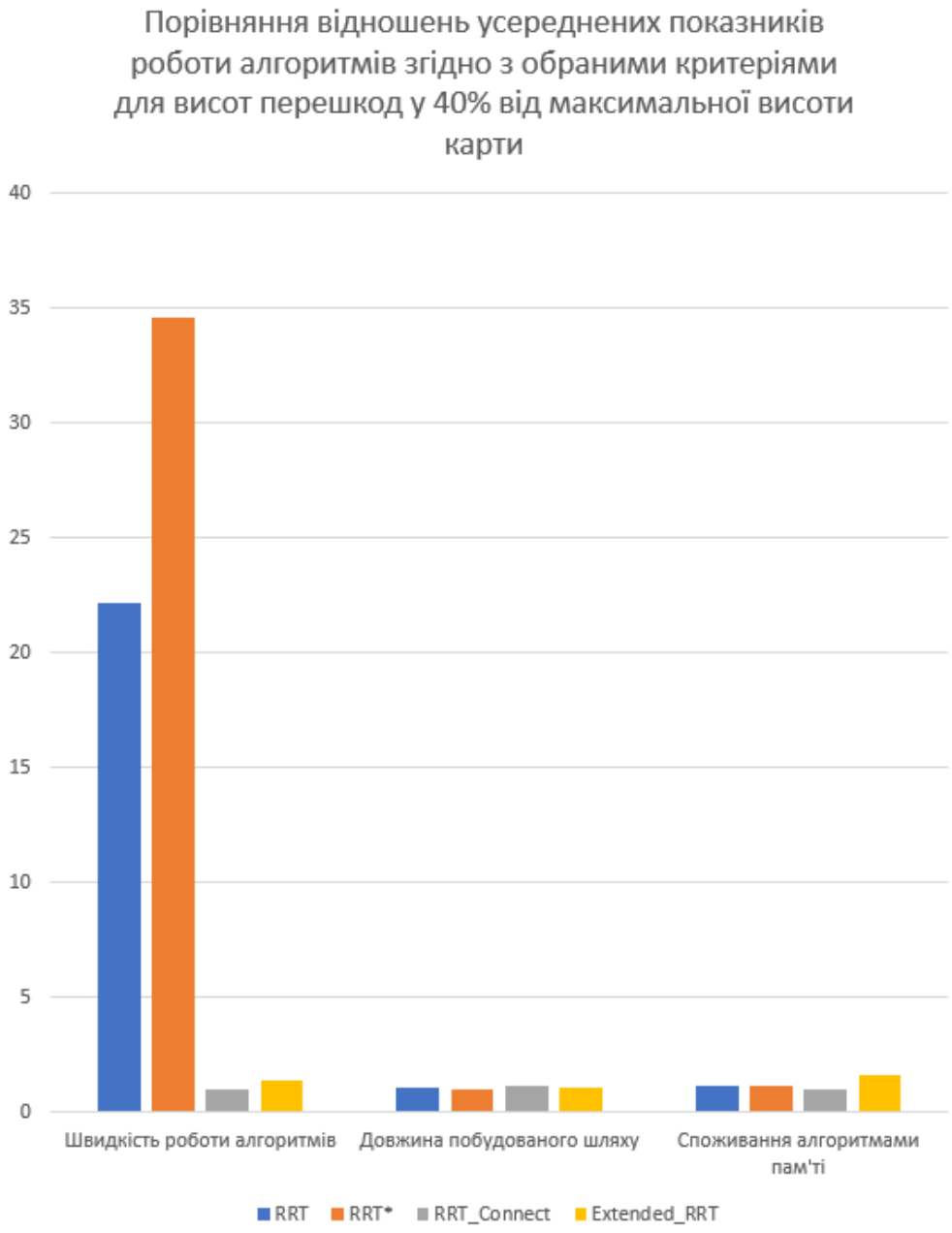

Рисунок 3.18 – Порівняння відношень усереднених показників роботи алгоритмів згідно з обраними критеріями для висот перешкод у 40 % від максимальної висоти карти

На рисунку 3.19 наведено порівняння відношень усереднених показників роботи алгоритмів пошуку шляху для маніпулятора за обраними критеріями, для висот перешкод у 60 % від максимальної висоти карти.

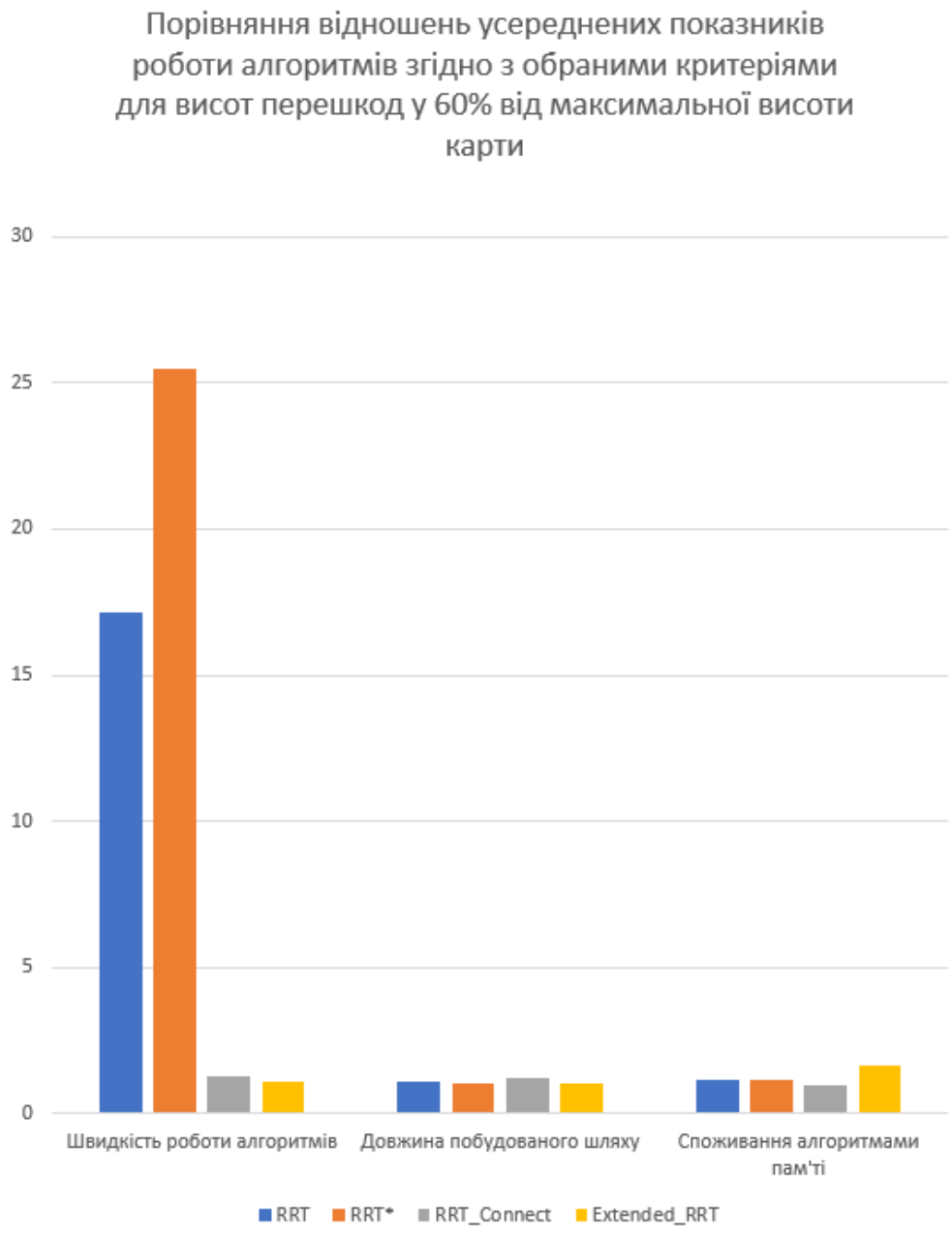

Рисунок 3.19 – Порівняння відношень усереднених показників роботи алгоритмів згідно з обраними критеріями для висот перешкод у 60 % від максимальної висоти карти

На рисунку 3.20 наведено порівняння відношень усереднених показників роботи алгоритмів пошуку шляху для маніпулятора за обраними критеріями, для висот перешкод у 80 % від максимальної висоти карти.

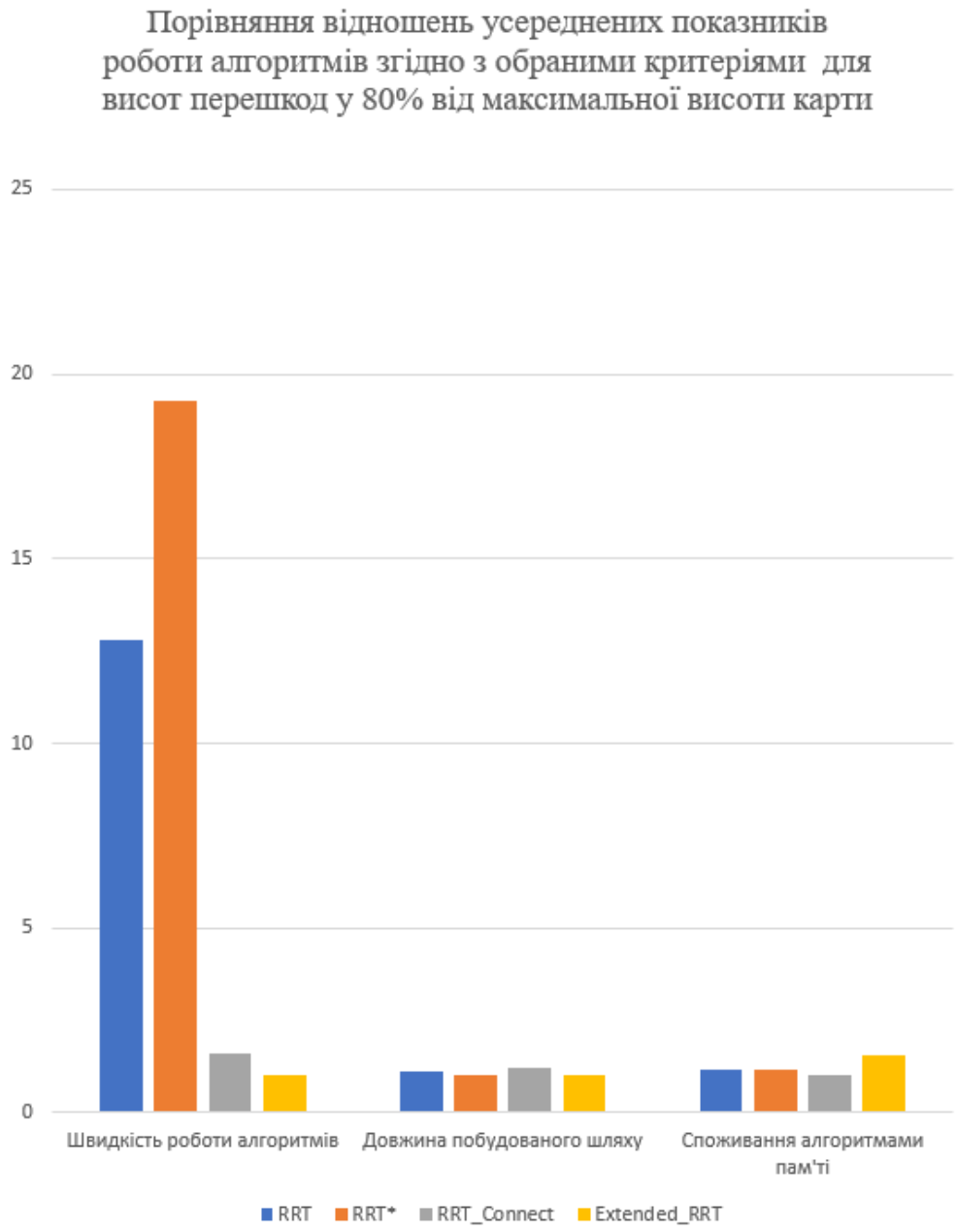

Рисунок 3.20 – Порівняння відношень усереднених показників роботи алгоритмів згідно з обраними критеріями для висот перешкод у 80 % від максимальної висоти карти

3.4 Висновки до третього розділу

В даному розділі для імітаційного моделювання роботи алгоритмів пошуку шляху для маніпуляторі в об’ємному просторі, було розроблено чотири карти з різною кількістю перешкод. Також для кожної окремої карти висота перешкод змінювалася. Було з’ясовано як впливає збільшення кількості перешкод та їх висота на показники алгоритмів, зроблено порівняння характеристик алгоритмів.

Усі розглянуті алгоритми пошуку шляху для маніпуляторі в об’ємному просторі пропонують, в залежності від кількості перешкод на карті та їх висоти, покращення одних характеристик за рахунок інших, що дає змогу створити СППР, яка буде мати змогу порекомендувати певний алгоритм, в залежності від обраної користувачем кількості перешкод на карті та їх висоти, а також в залежності від коефіцієнтів важливості, які користувач обирає для кожного з параметрів алгоритму.

4 РОЗРОБКА СППР

4.1 Вибір засобів реалізації

Python – це інтерпретована, інтерактивна, об’єктно-орієнтована мова програмування. Він містить модулі, винятки, динамічну типізацію, динамічні типи даних дуже високого рівня та класи. Він підтримує кілька парадигм програмування, окрім об’єктно-орієнтованого програмування, наприклад процедурне та функціональне програмування. Python поєднує в собі надзвичайну потужність із дуже чітким синтаксисом. Він має інтерфейси для багатьох системних викликів і бібліотек, а також для різних віконних систем і розширюється на C або C++. Його також можна використовувати як мову розширення для програм, яким потрібен програмований інтерфейс. Нарешті, Python портативний: він працює на багатьох варіантах Unix, включаючи Linux і macOS, а також на Windows [30].

Мова постачається з великою стандартною бібліотекою, яка охоплює такі області, як обробка рядків (регулярні вирази, Unicode, обчислення відмінностей між файлами), Інтернет-протоколи (HTTP, FTP, SMTP, XML-RPC, POP, IMAP), розробка програмного забезпечення, журналювання, профілювання, розбір коду та інтерфейси операційної системи (системні виклики, файлові системи, сокети TCP/IP). Також доступний широкий вибір сторонніх розширень .

Також для розробки СППР було обрано графічний інтерфейс Tkinter. Пакет tkinter («інтерфейс Tk») – це стандартний інтерфейс Python до інструментарію GUI Tcl/Tk [31]. Tkinter доступний на більшості платформ Unix, включаючи macOS, а також у системах Windows. Запуск tkinter із командного рядка має відкрити вікно, яке демонструє простий інтерфейс Tk, повідомляючи, що tkinter правильно встановлено у вашій системі, а також показує, яку версію Tcl/Tk встановлено. Tcl/Tk не є окремою бібліотекою, а складається з кількох окремих модулів, кожен з яких має окремі функції та власну офіційну документацію.

Tcl – це динамічна інтерпретована мова програмування, як і Python. Хоча її можна використовувати окремо як мову програмування загального призначення, найчастіше її вбудовують у додатки C як механізм сценаріїв або інтерфейс до набору інструментів Tk. Бібліотека Tcl має інтерфейс C для створення та керування одним або декількома екземплярами інтерпретатора Tcl, запуску команд і сценаріїв Tcl у цих екземплярах і додавання спеціальних команд, реалізованих у Tcl або C. Кожен інтерпретатор має чергу подій, а також є засоби для надсилання до нього подій та їх обробки.

Tk – це пакет Tcl, реалізований на C, який додає власні команди для створення та керування віджетами GUI. Кожен об'єкт Tk вбудовує власний екземпляр інтерпретатора Tcl із завантаженим у нього Tk. Віджети Tk дуже легко налаштовувати, хоча ціною застарілого вигляду. Tk використовує чергу подій Tcl для створення та обробки подій GUI.

Themed Tk (Ttk) – це нове сімейство віджетів Tk, які забезпечують набагато кращий вигляд на різних платформах, ніж багато класичних віджетів Tk.

4.2 Алгоритм роботи СППР

Як було зазначено у першому розділі – у якості метода прийняття рішень для розроблюваної СППР було обрано метод «згорти». Дані, які необхідні для використання цього методу були отримані у ході імітаційного моделювання в третьому розділі. Алгоритм роботи СППР зображено на рисунку 4.1.

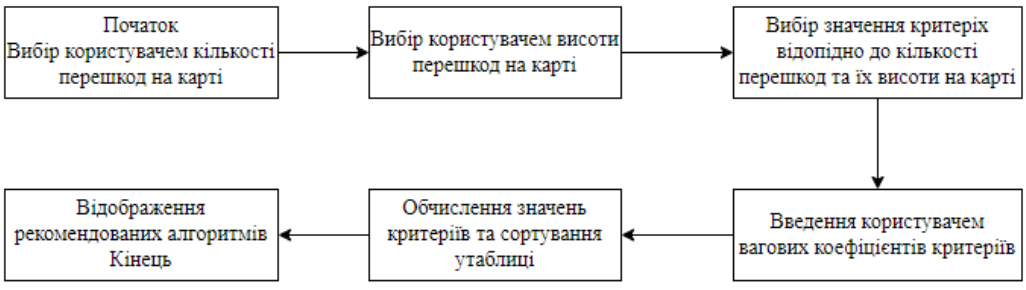

Рисунок 4.1 – Алгоритм роботи СППР

4.3 Тестування розробленої СППР

На рисунку 4.2 зображено зовнішній вигляд розробленої СППР

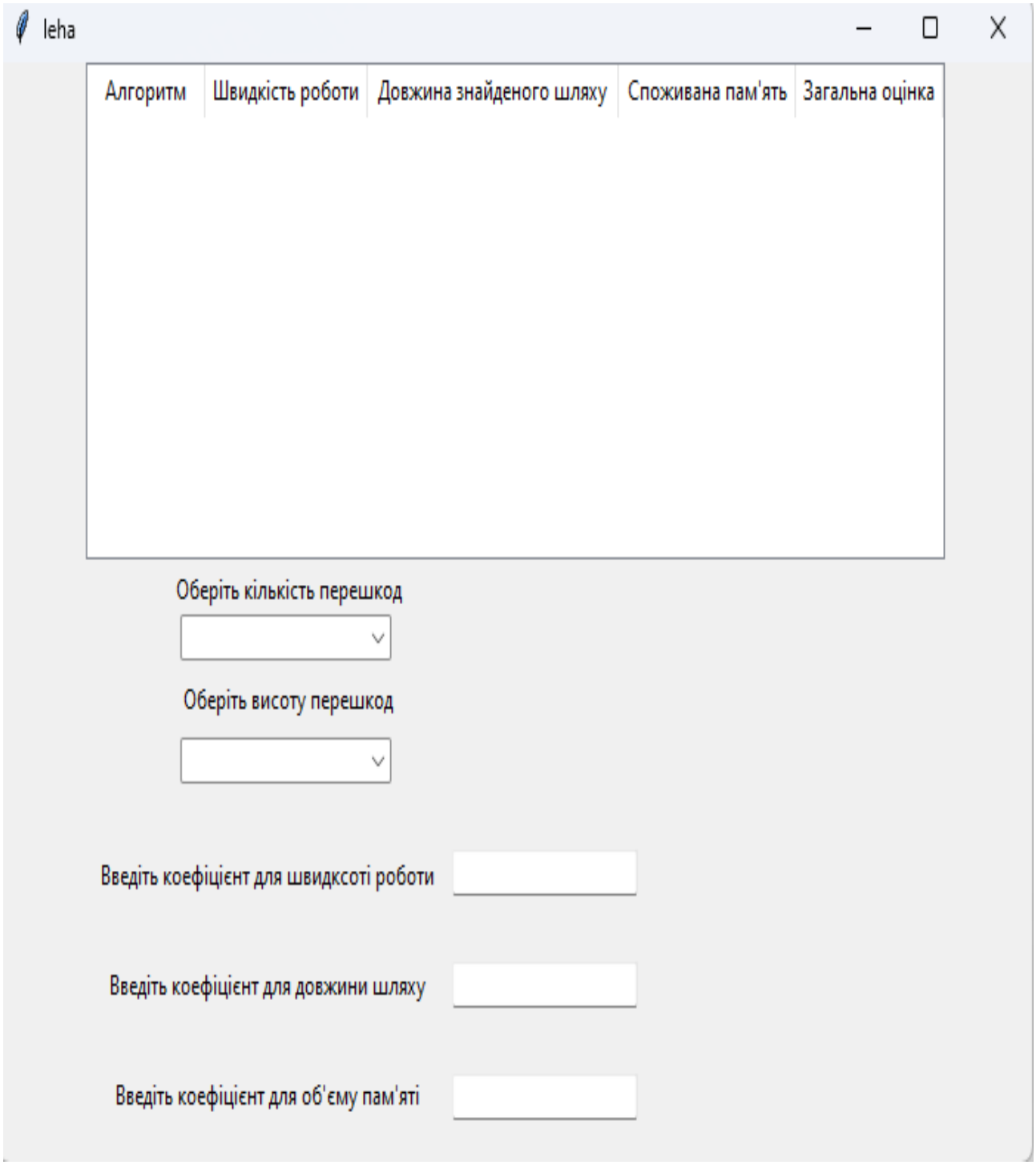

Рисунок 4.2 – Зовнішній вигляд створеної СППР

На рисунка 4.3 – 4.6 відображено результати роботи СППР для карти з однією перешкодою та різною висотою перешкод і різними індивідуальними ваговими коефіцієнтами для кожного критерію.

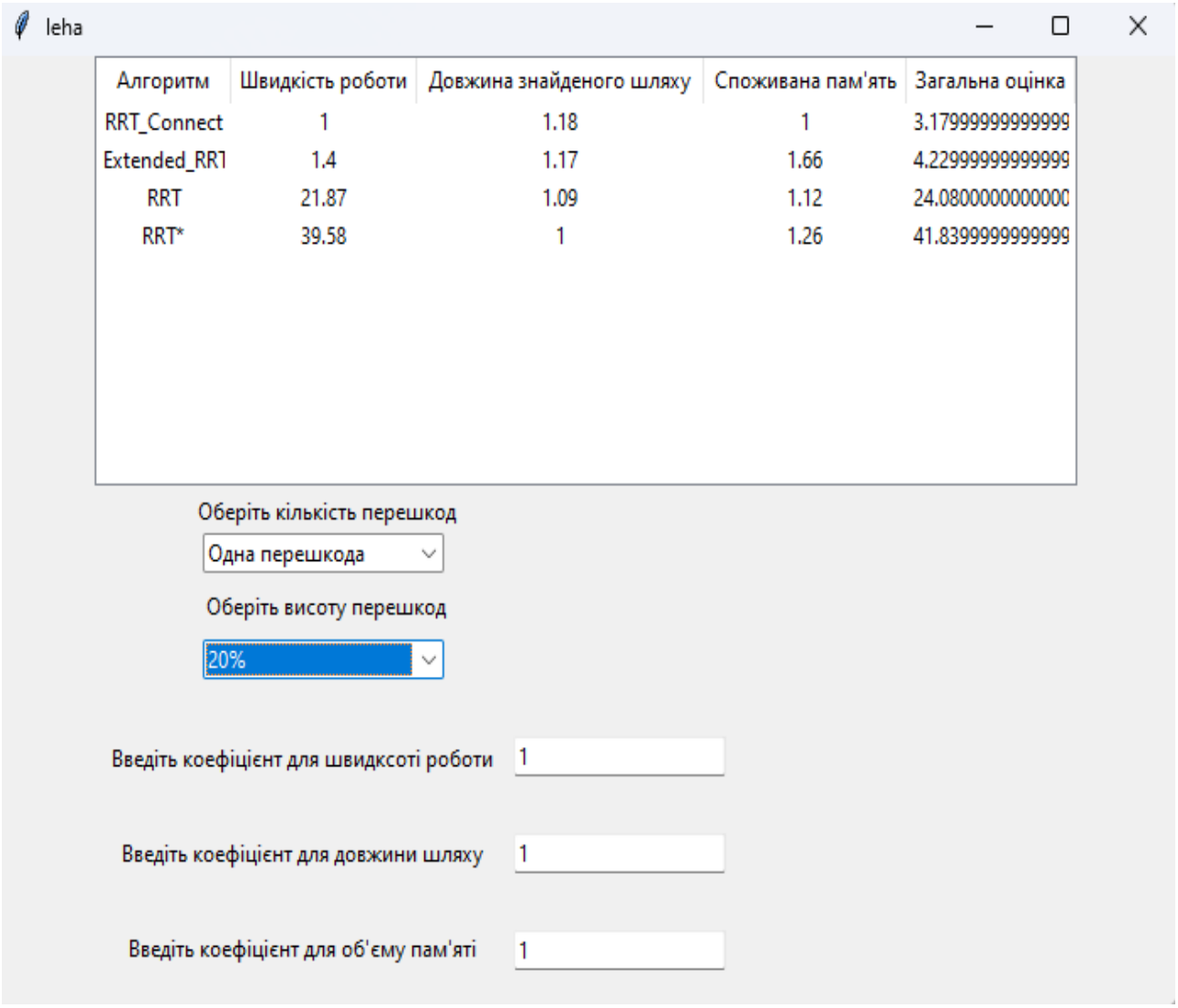

Рисунок 4.3 – Результати роботи СППР для карти з однією перешкодою та висотою перешкоди у 20 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.3 можна побачити, що для карти з однією перешкодою і її висотою у 20 % від максимальної висоти карти, при обраних вагових

коефіцієнтах в одиницю СППР рекомендує алгоритми RRT-Connect та Extended-RRT.

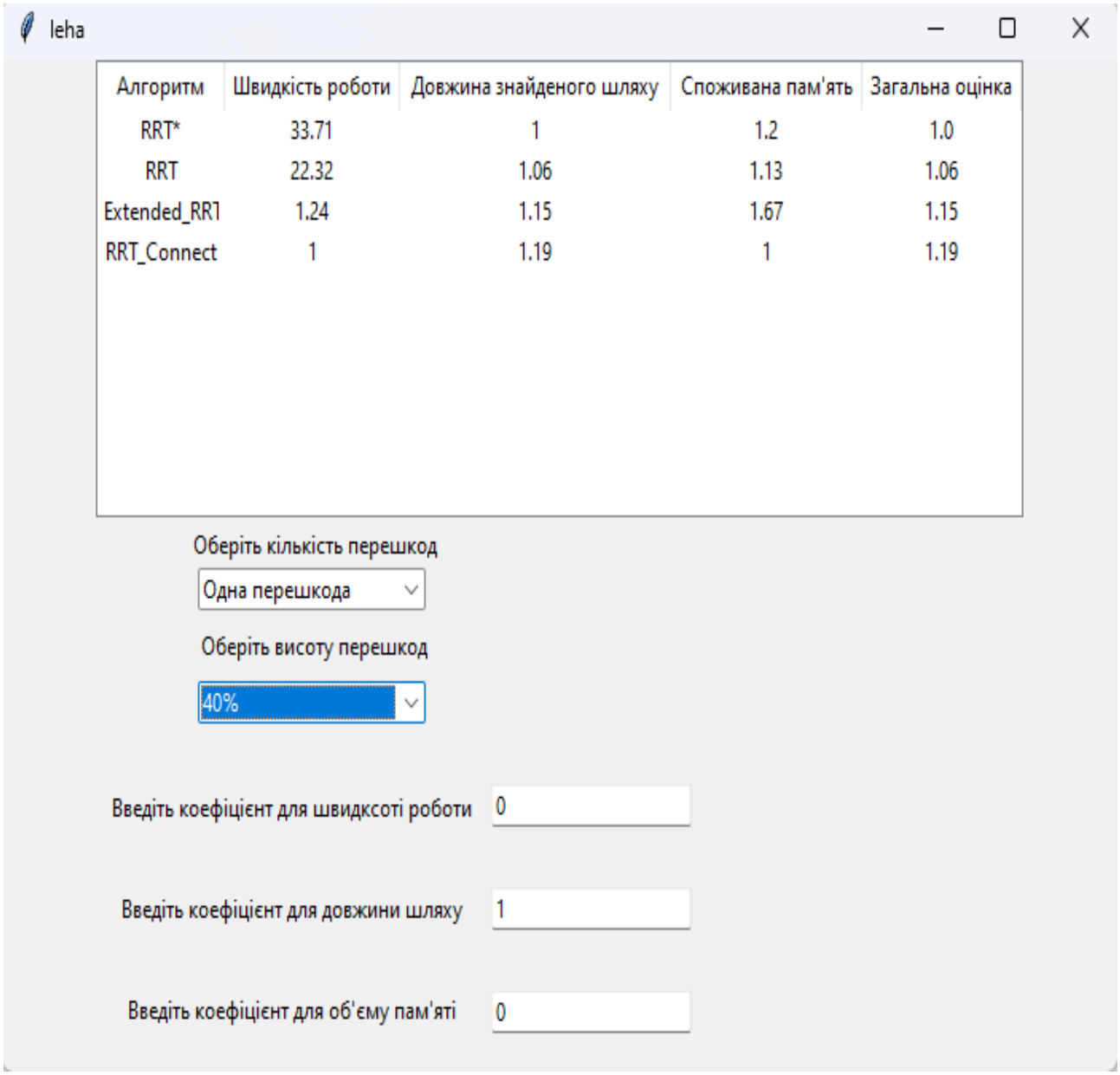

Рисунок 4.4 – Результати роботи СППР для карти з однією перешкодою та висотою перешкоди у 40 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.4 можна побачити, що при висоті перешкоди у 40 % від максимальної висоти карти, для карти з однією перешкодою та опираючись

тільки на коефіцієнт довжини шляху, розроблена СППР рекомендує алгоритми RRT^* та стандартний RRT.

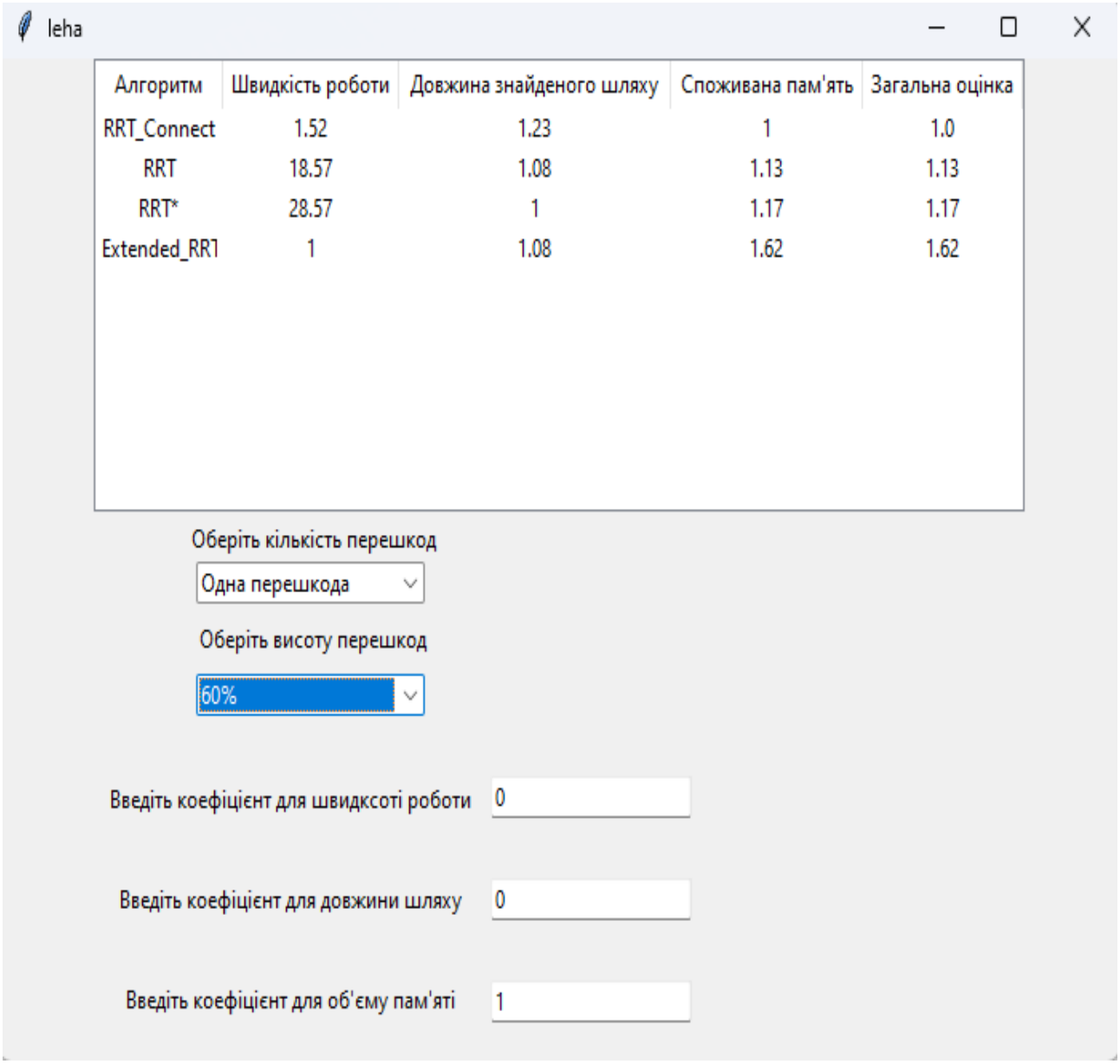

Рисунок 4.5 – Результати роботи СППР для карти з однією перешкодою та висотою перешкоди у 60 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.5 можна побачити, що для карти з однією перешкодою та висотою перешкоди у 60 % від максимальної висоти карти, опираючись тільки

на коефіцієнт об’єму споживаної пам’яті, розроблена СППР рекомендує алгоритми RRT-Connect та стандартний RRT.

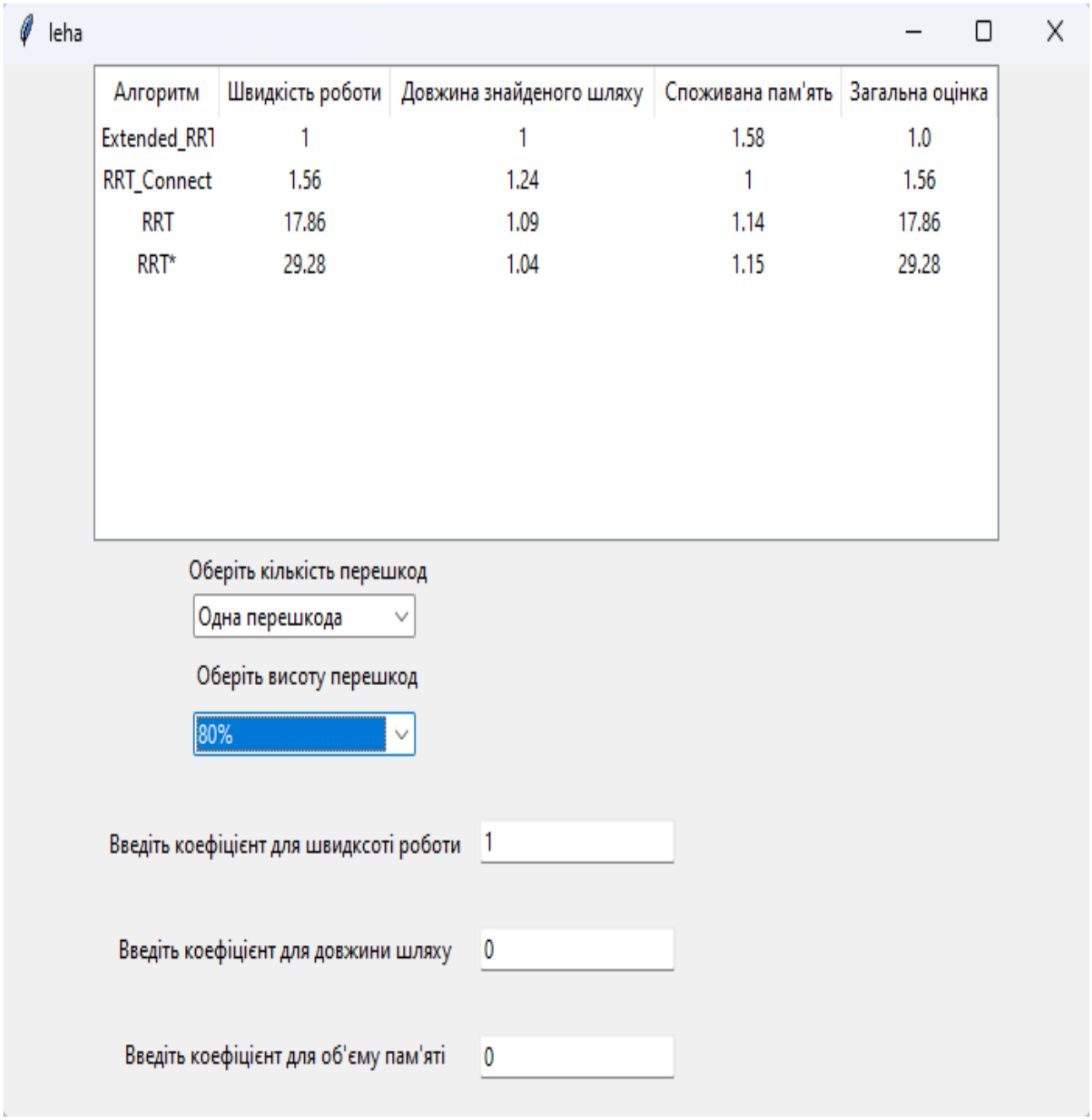

Рисунок 4.6 – Результати роботи СППР для карти з однією перешкодою та висотою перешкоди у 80 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

З рисунку 4.6 видно, для карти з однією перешкодою, висотою у 80 % від максимальної висоти карти, опираючись тільки на коефіцієнт швидкості роботи алгоритму, розроблена СППР рекомендує алгоритми Extended-RRT та RRT-Connect.

На рисунка 4.7 – 4.8 відображено результати роботи СППР для карти з двома перешкодами та різною висотою перешкод і різними індивідуальними ваговими коефіцієнтами для кожного критерію.

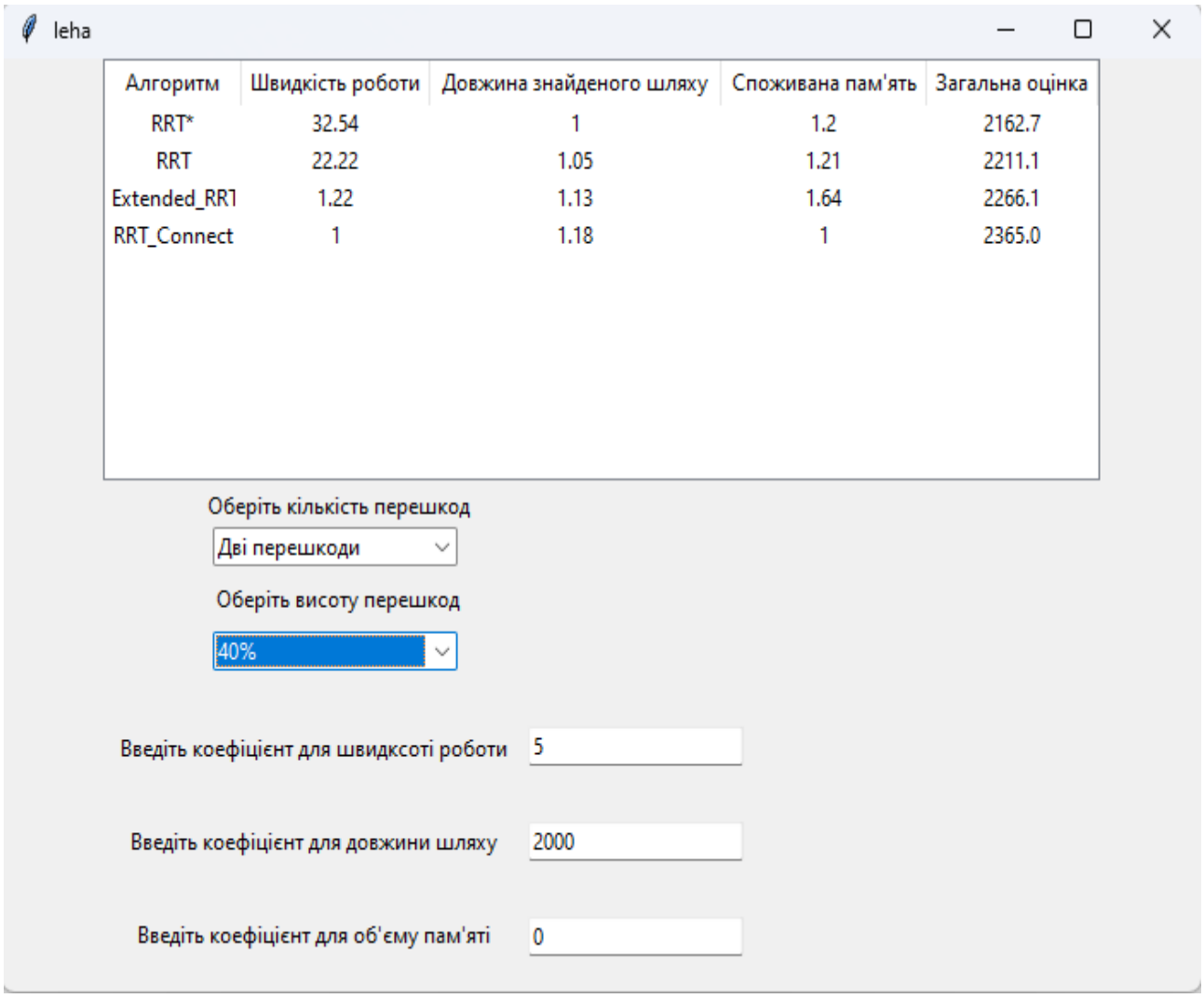

Рисунок 4.7 – Результати роботи СППР для карти з двома перешкодами та висотою перешкоди у 40 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.7 видно, що для карти з двома перешкодами та висотою перешкод у 40 % від максимальної висоти карти, опираючись більшою мірою на ваговий коефіцієнт довжини шляху ніж на ваговий коефіцієнт швидкості роботи алгоритму, розроблена СППР рекомендує алгоритми RRT^* та стандартний RRT.

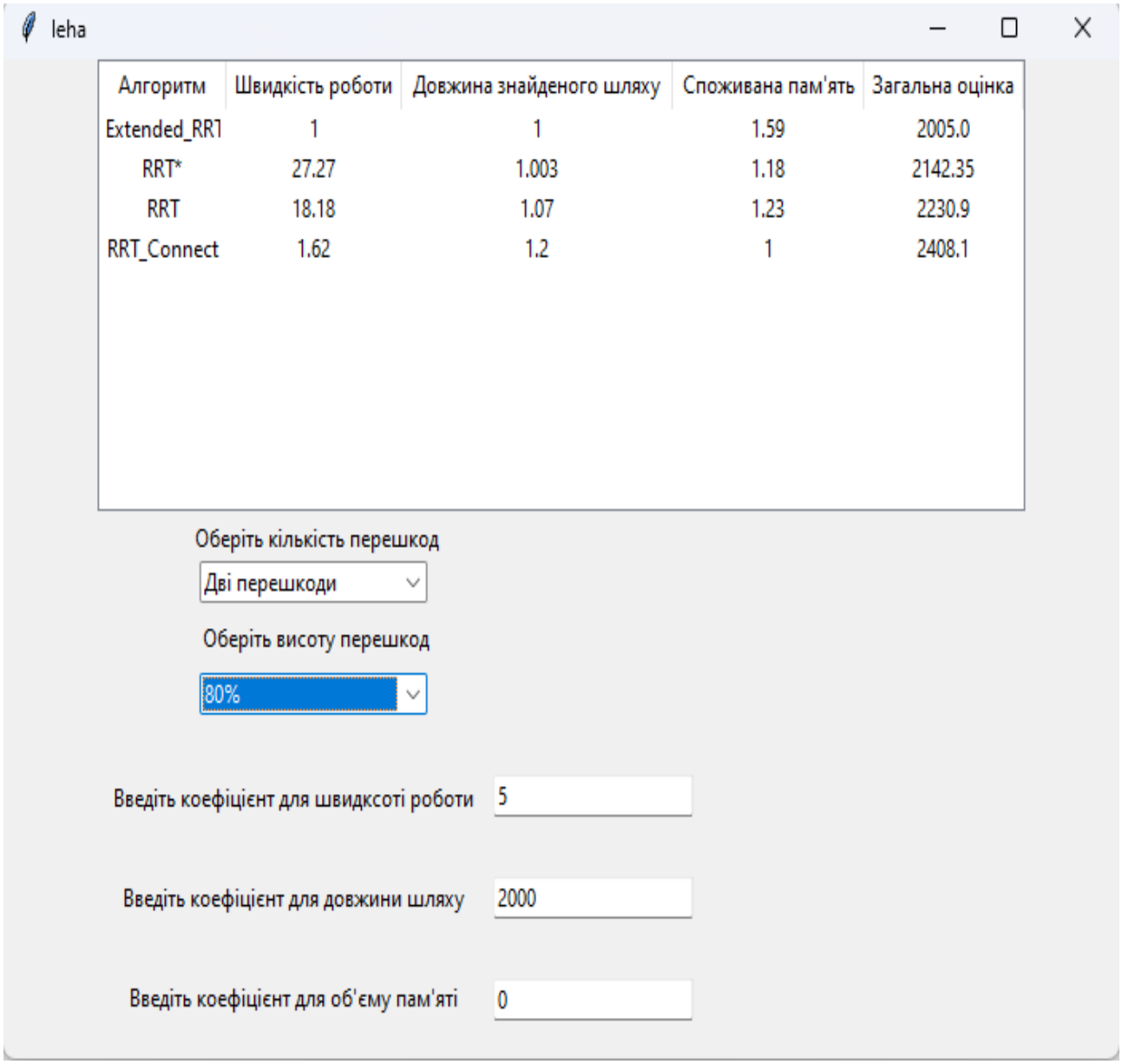

Рисунок 4.8 – Результати роботи СППР для карти з двома перешкодами та висотою перешкоди у 80 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

З рисунку 4.8 видно, що для карти з двома перешкодами і висотою перешкод у 80 % від максимальної висот карти опираючись на ті самі вагові коефіцієнти, розроблена СППР рекомендує вже алгоритми Extended-RRT та RRT^*.

На рисунках 4.9 – 4.10 відображено результати роботи СППР для карти з трьома перешкодами та різною висотою перешкод і різними індивідуальними ваговими коефіцієнтами для кожного критерію.

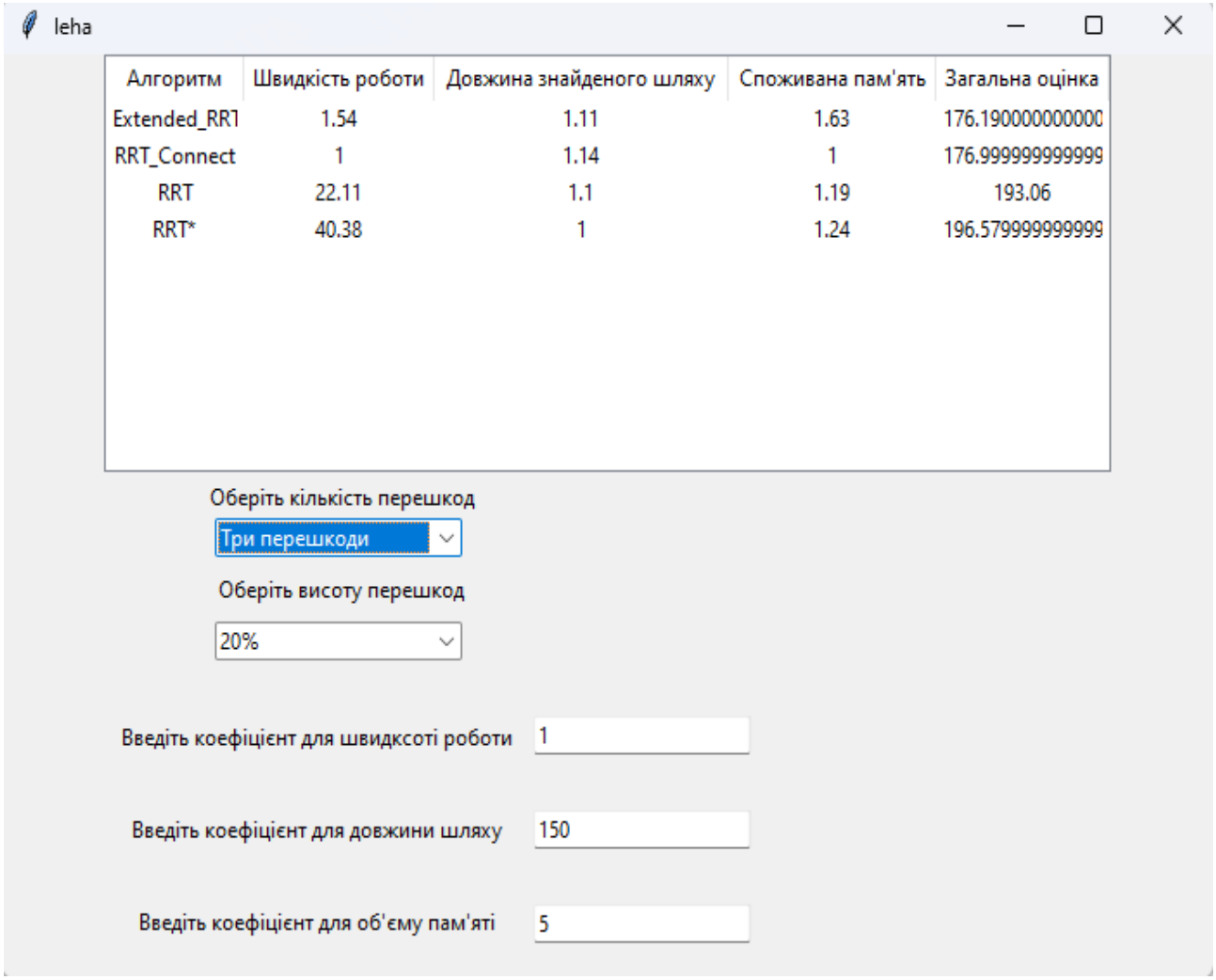

Рисунок 4.9 – Результати роботи СППР для карти з трьома перешкодами та висотою перешкоди у 20 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.9 видно, що для карти з трьома перешкодами та висотою перешкод у 20 % від максимальної висоти карти опираючись на ваговий коефіцієнт довжини шляху більше ніж на ваговий коефіцієнт об’єму споживаної пам’яті, розроблена СППР рекомендує алгоритми Extended-RRT та RRT-Connect.

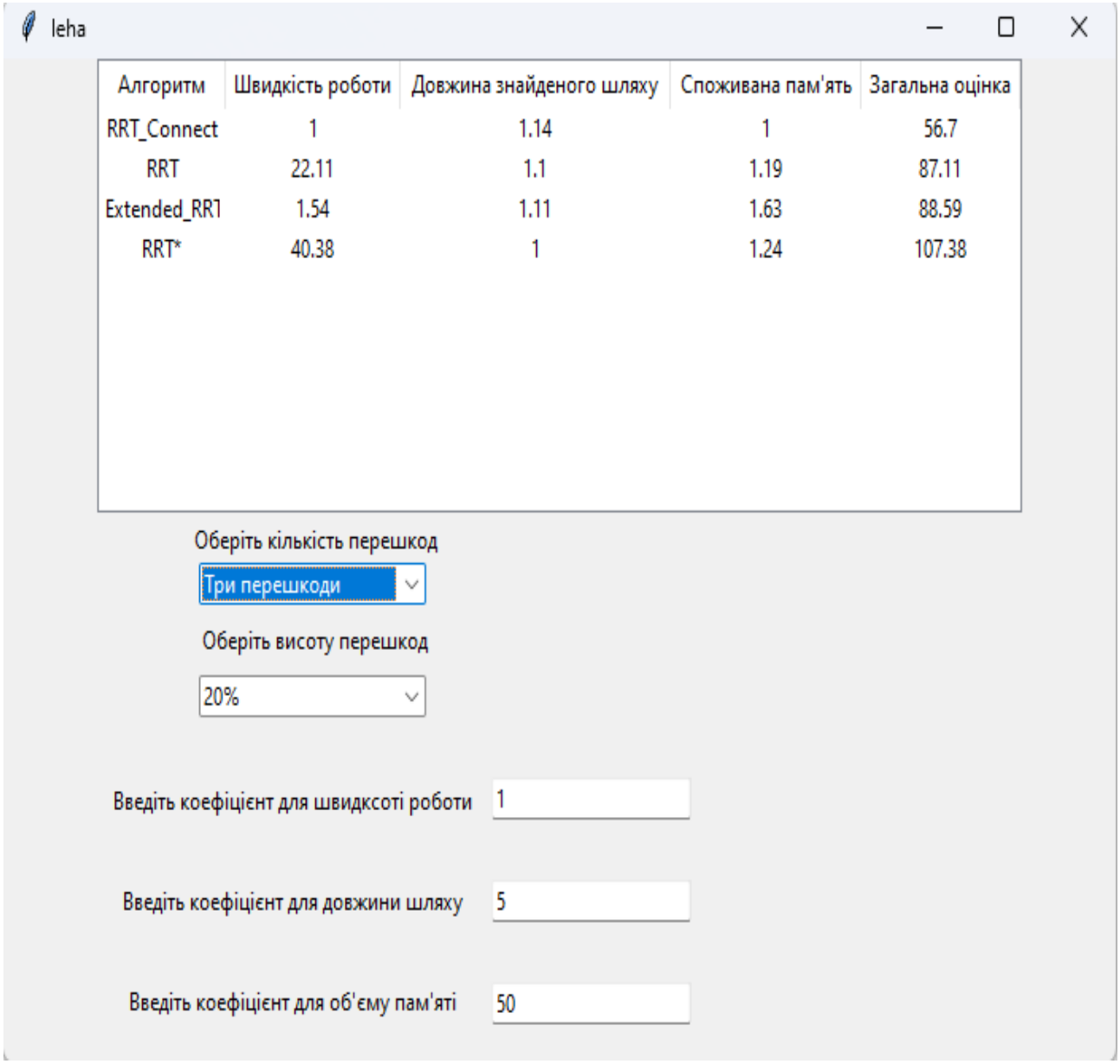

Рисунок 4.10 – Результати роботи СППР для карти з трьома перешкодами та висотою перешкоди у 20 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.10 видно, що для карти з трьома перешкодами і такою ж вистою перешкод, але опираючись все на ваговий коефіцієнт об’єму споживаної пам’яті в більшій мірі ніж на ваговий коефіцієнт довжини побудованого шляху, розроблена СППР рекомендує алгоритми RRT-Connect та RRT.

На рисунка 4.11 – 4.12 відображено результати роботи СППР для карти з чотирма перешкодами та різною висотою перешкод і різними індивідуальними ваговими коефіцієнтами для кожного критерію.

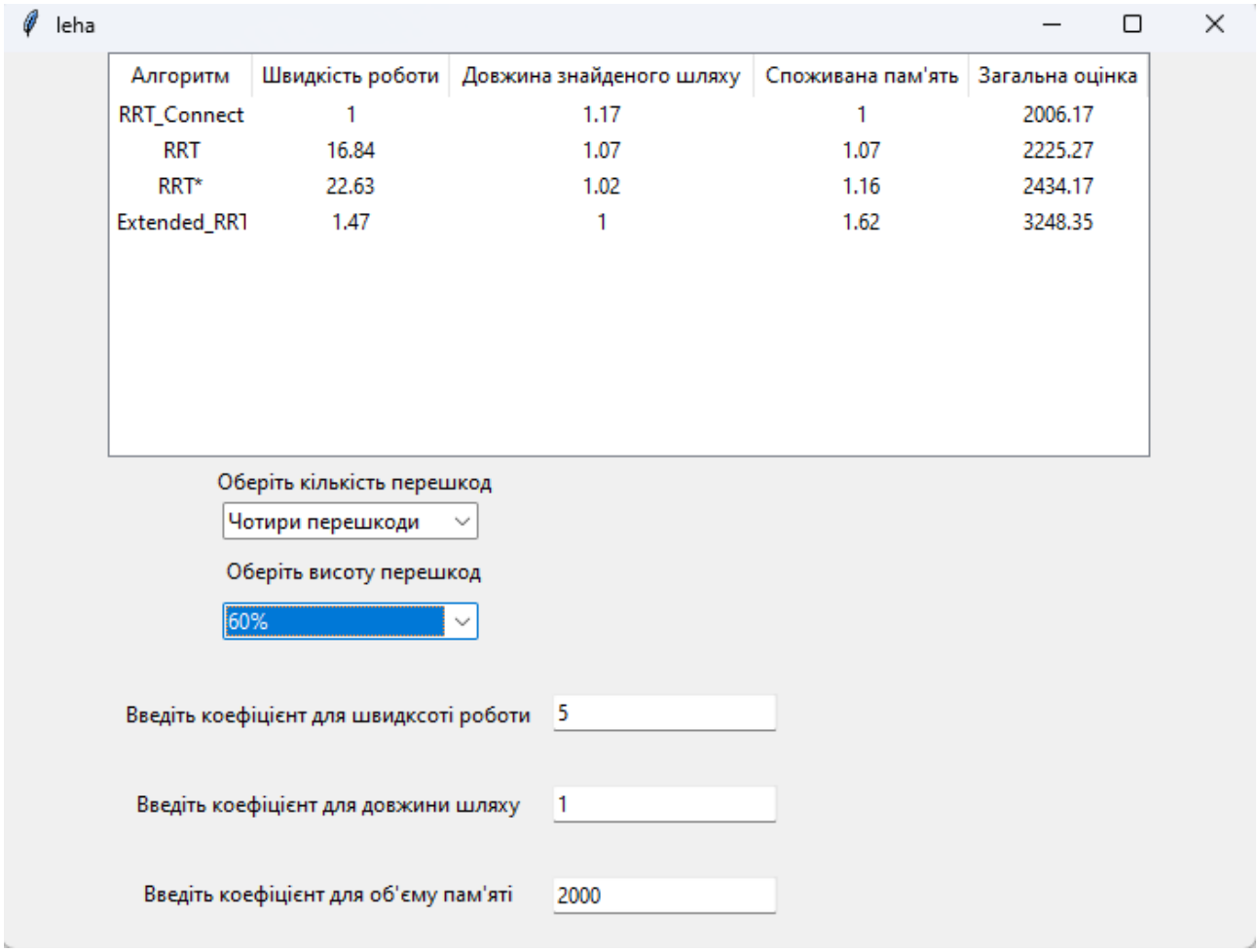

Рисунок 4.11 – Результати роботи СППР для карти з чотирма перешкодами та висотою перешкоди у 60 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.11 видно, що для карти з чотирма перешкодами і висотою перешкод у 60 % від максимальної висоти карти, опираючись на ваговий коефіцієнт об’єму споживаної пам’яті в більшій мірі ніж на коефіцієнт швидкості роботи, розроблена СППР рекомендує алгоритми RRT-Connect та RRT.

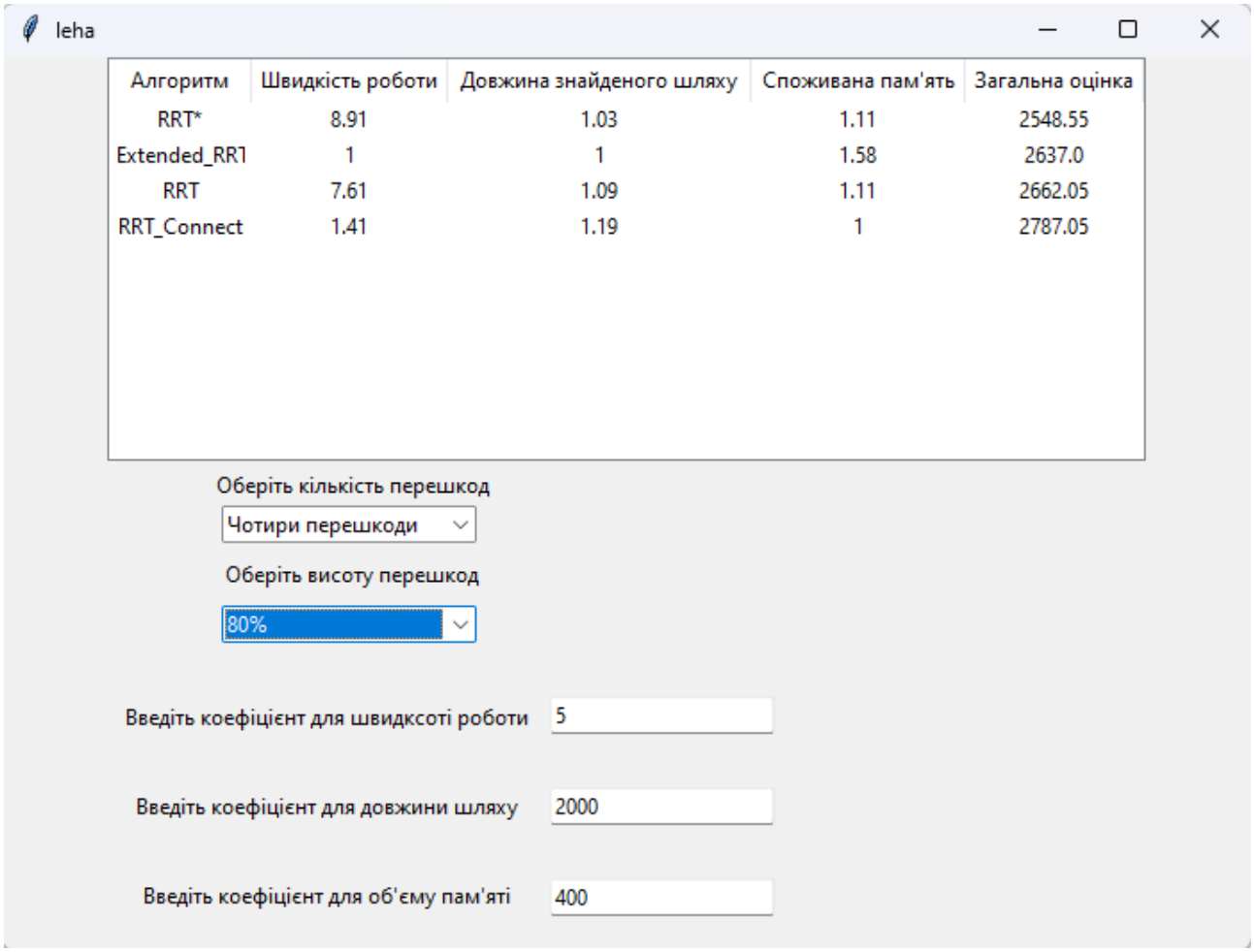

Рисунок 4.12 – Результати роботи СППР для карти з чотирма перешкодами та висотою перешкоди у 80 % від максимальної висоти карти і різними індивідуальними ваговими коефіцієнтами для кожного критеріїв

На рисунку 4.12 видно, що для карти з чотирма перешкодами та висотою перешкод у 80 % від максимальної висоти карти, опираючись спершу на ваговий коефіцієнт довжини шляху, потім на ваговий коефіцієнт об’єму споживаної

пам’яті і врешті на ваговий коефіцієнт швидкості роботи, розроблена СППР рекомендую алгоритми RRT^* та Extended-RRT.

4.4 Висновки за розділом

В даному розділі було розроблено алгоритм функціонування СППР, обрано засоби її створення, і реалізовано СППР у вигляді програмного додатку. Проведено тестування роботи розробленої СППР для карт з різною кількістю перешкод, різною висотою перешкод та різними ваговими коефіцієнтами для кожного критерію, які індивідуально обирає користувач.

5 ОХОРОНА ПРАЦІ

Розробка програмного забезпечення проводиться в спеціальному приміщенні. Воно має параметри: п’ять робочих місць; один ПК (с рідкокристалічними монітором). Електрична мережа приміщення має такі характеристики: трифазна чотирьохдротова мережа напругою 380/220 В змінного струму, частота 50 Гц, з глухозаземленою нейтраллю. Функціональна схема одного робочого місця представлена на рисунку 4.1.

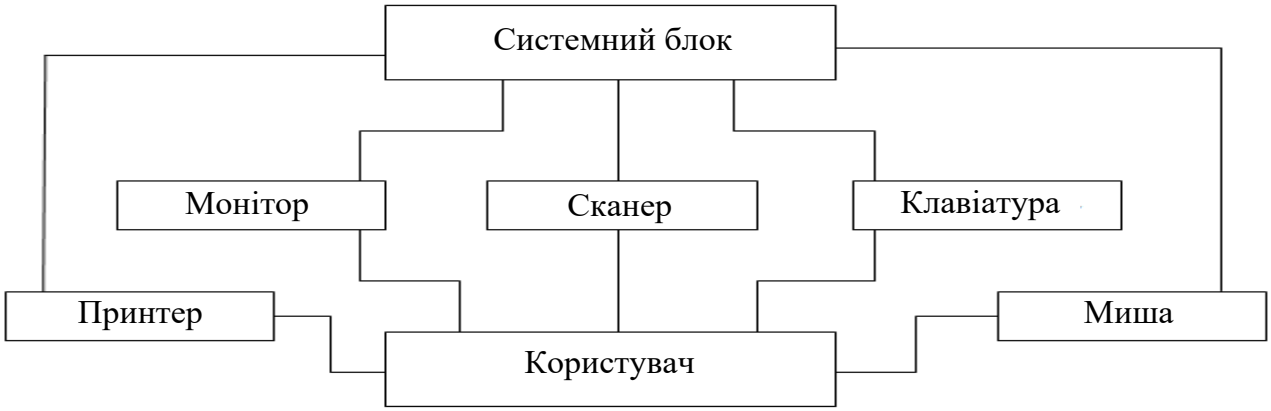

Рисунок 5.1 – Функціональна схема робочого місця

Приміщення відділу для розробки програмного забезпечення відноситься до приміщень без підвищеної небезпеки поразки людей електричним струмом згідно з НПАОП 40.1–1.21:98, так як немає умов створюють підвищену або особливу небезпеку. У приміщенні знаходиться електрощит, на якому встановлено пристрій струмового захисту. Всі розетки мають застережливий напис «220». З робітниками проводяться інструктажі з охорони праці відповідно до НПАОП 0.00-4.12:2005 [32].

Згідно з вимогами НПАОП 0.00–4.12:2005 необхідно проводити вступний, первинний на робочому місці, повторний, а при необхідності – позаплановий і цільовий інструктажі.

Схема занулення представлена на рисунку 5.2, параметри цієї схеми відображені в таблиці 5.1.

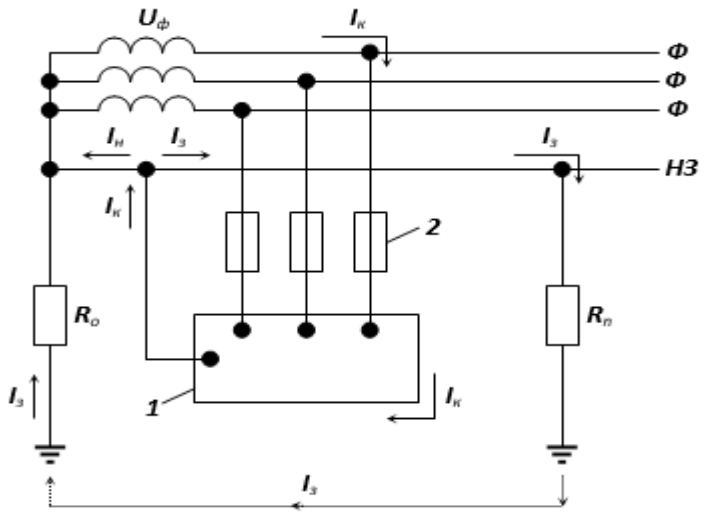

Рисунок 5.2 – Принципова схема занулення в трифазній мережі до 1000В

Таблиця 5.1 – Параметри схеми занулення в трифазній мережі до 1000 В

1	корпус електроустановки
2	апарати захисту від струмів короткого замикання (КЗ)
Ф	фазний провід
НЗ	нульовий захисний провідник
$R_о$	опір заземлення нейтрали обмотки джерела струму
$R_п$	опір повторного заземлення нульового захисного провідника
$I_к$	ток КЗ
$I_н$	частина струму КЗ, що протікає через нульовий захисний провідник
$I_з$	частина струму КЗ, що протікає через землю

5.1 Висновки до п’ятого розділу

Проведено огляд робочого місця, визначенні основні параметри , наведено необхідні вимоги, визначенні можливі фактори небезпеки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз технічного завдання, а саме проаналізовано існуючі дослідження та виявлено, що по сумі критеріїв для побудови шляху для маніпулятора в об’ємному просторі, планування на основі вибірки є найбільш оптимальним. У наведених дослідження метод був представлений алгоритмами ймовірнісного методу дослідження випадкових карт (PRM) та алгоритмом швидкого дослідження випадкових дерев (RRT). З двох наведених, найкращим був обраний алгоритм швидкого дослідження випадкових дерев, тому було прийнято рішення проаналізувати сімейство алгоритмів RRT, визначити їх параметри та на основі отриманих даних створити СППР для задач будування шляху в об’ємному просторі для маніпулятора. Також у розділі були розглянуті системи підтримки прийняття рішень, їх структура та методи розв’язання оптимізаційних задач.

Другий розділ дипломної роботи було присвячено детальному розгляду та аналізу сімейства алгоритмів швидкого дослідження випадкових дерев, а саме RRT, RRT^*, RRT-Connect, Extended-RRT, розглянуто механізми їх функціонування.

Третій розділ кваліфікаційної роботи було присвячено проведенню імітаційного моделювання задля отримання результатів роботи кожного алгоритму за певних критеріїв для різного навколишнього середовища, щоб в подальшому використати ці показники для створення СППР.

Було створено об’ємну чотири варіанти карти для пошуку шляху алгоритмами з розмірами 20x20x10 з різною кількістю перешкод. Також для кожного варіанту карти змінювалася висота перешкод відносно максимальної висоти карти. Було проаналізовано швидкість роботи алгоритмів, довжину побудованого ними маршруту та об’єм споживаної пам’яті для кожної варіації карти, щоб в подальшому використати ці показники для створення СППР.

Для кожного алгоритму та варіації карти було візуалізовано знайдені алгоритмами шляхи, що дозволяє додатково побачити різницю між ними.

У четвертому розділі було створено СППР базуючись на даних отриманих в третьому розділі та обрані засоби для реалізації створеного алгоритму СППР. Було проведено тестування СППР та визначено коректність функціонування.

Базуючись на виконаній роботі та отриманим в результаті даним, пропонується створити експериментальний модуль, з'єднаний з маніпулятором, та модуль запису результатів. Модуль запису результатів, має обробляти дані експериментального модуля та вносити результати до бази знань СППР. Також для збільшення точності СППР, бо побудований алгоритмами маршрут є досить випадковим і може доволі сильно відрізнятися для кожної ітерації, пропонується зробити якомога більшу кількість прогонів кожного з алгоритмів для кожної окремо взятої карти місцевості, тому перед проведенням таких експериментів рекомендується проаналізувати необхідність і рентабельність створення настільки точної СППР. Стосовно СППР створеної в ході дипломної роботи, використовуючи її можна отримати відносно точну рекомендацію щодо вибору алгоритму побудови шляху в об'ємному просторі для маніпулятора в залежності від параметрів карти місцевості.

Науковою новизною даної роботи є аналіз характеристик алгоритмів RRT, RRT^*, RRT-Connect, Extended-RRT, в залежності від параметрів конфігурації карти місцевості та розробка СППР на основі отриманих даних, за допомогою якої, в загальному випадку, можна обрати оптимальний алгоритм для побудови маршруту о об'ємному просторі для маніпулятора, в залежності від обраних критеріїв.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП “УкрНДНЦ”. 2016. 30 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп’ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами»,«Комп’ютерно-інтегровані технологічні процеси і виробництва», «Комп’ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2021. 55 с.

3. Будова та система керування мобільної платформи на базі робота павука / Бєлов П. О., Мандрікін М. С., Катков О. В., Цимбал О. М. Харків: Topical issues of modern science, society and education. Proceedings of the 6th International scientific and practical conference, 2021. С. 343.

4. Review on Development of Industrial Robotic Arm / Rahul Gautam, Ankush Gedam, Ashish Zade, Ajay Mahawadiwar. Internationl Research Journal of Engineering and Technology, 2017. С. 20 – 23.

5. Modeling, identification and minimum length integral sliding mode control of a 3-DOF cartesian parallel robot by considering virtual flexible links / A. Beiranvand, A. Kalhor, M. Masouleh. Mechanism and Machine Theory, 2021. С. 45 – 56.

6. Control of Polar Robot Manipulator Subjected to Harmonic Excitation Using Low Cost Controller / M. Mabrouk, A. Abdel-Hamid. Proceedings of the World Congress on Engineering, 2019. С. 56 – 62.

7. Dynamic Processes of Loader Cranes Manipulators with Excessive Backlashes and Elastic Damping in Their Hinges / A. Lagerev, I. Lagrev. Periodica Polytechnica Mechanical Engineering, 2020. С. 117 – 123.

8. Design, construction and control of a SCARA manipulator with 6 degrees of

freedom / C. Urrea, J. Cortes, J Pascal. Journal of applied research and technology, 2016. С. 74 – 91.

9. Trends and challenges in robot manipulation / Aude Billard, Danica Kragic. Science, 2019. С. 82 – 94.

10. A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots / A. Hentout, A. Maoudj, M. Aouache. Artificial Intelligence Review, 2022. С. 117 – 125.

11. Understanding Dijkstra’s algorithm / A. Javaid. CompTIA, 2013. С. 42 – 53.

12. D-star Algorithm Modification / S. Kadry, G. Alferov, V. Fedorov. International Journal of Online & Biomedical Engineering, 2020. С. 51 – 62.

13. Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment / G. Tang, C. Tang, C. Claramunt, Xiong Hu, Peipei Zhou – IEEE, 2021. С. 55 – 72.

14. Sampling-Based Robot Motion Planning: A Review / M. Elbanhawi, M. Simic – IEEE, 2014. С. 41 – 62.

15. Sampling-Based Methods for Motion Planning with Constraints / Z. Kingston, M. Moll, Lydia E. Kavraki. Annual Reviews, 2018. С. 54 – 72.

16. Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method / Fei Yan, Yi-Sha Liu, Ji-Zhong Xiao, 2014. International journal of automatization and computing. С. 20 – 32.

17. A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning / F. Neri, J. Campus. Integrated Computer-Aided Engineering, 2020. С. 51 – 68.

18. A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation / J. Mohanta. Applied Soft Computing, 2019. С. 61 –74.

19. An Adaptive Rapidly-Exploring Random Tree / B. Li, B. Chen – Journal of Automatica Sinica, 2021. С. 72 – 83.

20. Системи і методи підтримки прийняття рішень / П. І. Бідюк, О. Л. Тимощук, А. Є. Коваленко, Л. О. Коршевнюк. Київ: КПІ ім. Ігоря Сікорського, 2022 С. 51 – 72.

21. Rapidly-Exploring Random Trees: A New Tool for Path Planning / S. LaValle – Mathematics, 1998. С. 42 – 52.

22. A Heuristic Rapidly-Exploring Random Trees Method for Manipulator Motion Planning / C. Yuan, W. Zhang, G. Liu – IEEE, 2019. С. 55 – 63.

23. Parallel RRT architecture design for motion planning / S. Xiao, N. Bergmann, A.Postula. International Conference on Field Programmable Logic and Applications, 2017. С. 71 – 92.

24. Sampling-based Algorithms for Optimal Motion Planning / S. Karaman, E. Frazzolo – The International Journal of Robotics Research, 2011. С. 51 – 63.

25. RT-Connect: Faster, asymptotically optimal motion planning / S. Klemm, A. Hermann, A. Roennau. International Conference on Robotics and Biomimetics, 2015. С. 23 – 55.

26. RRT-connect: An efficient approach to single-query path planning / J. Kuffner, S. LaValle. International Conference on Robotics and Automation, 2000. С. 17 – 32.

27. Efficient Robot Motion Planning Using Bidirectional-Unidirectional RRT Extend Function / J. Wang, W. Chi. Transactions on Automation Science and Engineering, 2021. С. 55 – 63.

28. Real-Time Randomized Path Planning for Robot NavigationReal-Time Randomized Path Planning for Robot Navigation / J. Bruce, M. Veloso – RoboCup 2002: Robot Soccer World Cup VI, 2002. С. 32 – 42.

29. How to Build Valid and Credible Simulation Models / Averill M. Law – Winter Simulation Conference, 2019. С. 55 – 71.

30. General Python FAQ. URL: https:docs.python.org/3/faq/general.html (дата звернення : 10.10.2022).

31. tkinter – Python interface to Tcl/Tk. URL: https:docs.python.org/3/library/tkinter.html (дата звернення : 10.10.2022).

32. Перелік робіт з підвищеною небезпекою (НПАОП 0.00-4.12-2005). URL: https://zakon.rada.gov.ua/laws/show/z0232-05 (дата звернення 10.10.2022).