Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
(повна назва)

Кафедра медіаінженерії та інформаційних радіоелектронних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*
(позначення документа)

Розробка програмно-апаратних засобів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод
(тема)

Виконав:
студент 2 курсу, групи *МІм-17-1*
Момот І.В.
(прізвище, ініціали)

Спеціальність
172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Медіаінженерія*
(повна назва освітньої програми)

Керівник *проф. Колендовська М.М.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Карташов В.М.
(підпис) (прізвище, ініціали)
2021 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформації
Кафедра Медіаінженерії та інформаційних радіоелектронних систем

Рівень вищої освіти другий (магістерський)
Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма "Медіаінженерія"

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові Момот Ігор Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмно-апаратних засобів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод
затверджена наказом по університету від " 08 " 11 2021 р. № 1676
2. Термін подання студентом роботи 12.12.2021 р.
3. Вихідні дані до проекту (роботи)
1. Розробити функціональну схему системи навігації мобільних роботів
2. Розробити модель розпізнавання перешкод мобільним роботом
3. Розробити алгоритм подолання перешкод мобільним роботом

4. Перелік питань, що потрібно опрацювати в роботі
ВСТУП
1. Аналіз існуючих програмно-апаратних засобів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод
2. Обгрунтування використання штучного інтелекту для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод
3. Аналіз використання ігрових двигунів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод
4. Розробка моделей для дослідження поведінки мобільних роботів при розпізнаванні перешкод
ВИСНОВКИ
ПЕРЕЛІК ПОСИЛАНЬ
ДОДАТКИ

5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій:
1. Постановка задачі; 2. Схема методу вимірювань координат; 3. Сфера застосування; 4. Актуальність; 5. Структура мобільного робота 6. Визначення понять; 7. Основні оптичні прилади та системи навігації; 8. Функціональна схема системи; 9. Розробка моделі оптичної системи навігації мобільних роботів; 10. Аналіз

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз існуючих програмно-апаратних засобів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод	08.11.21–15.11.20	
2.	Обгрунтування використання штучного інтелекту для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод	15.11.21–20.11.21	
3.	Аналіз використання ігрових двигунів для дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод	21.11.21–27.11.21	
4.	Розробка моделей для дослідження поведінки мобільних роботів при розпізнаванні перешкод	28.11.21–02.12.21	
5.	Графічна частина роботи	25.11.21–02.12.21	
6.	Перевірка керівником	02.12.21-03.12.21	
7.	Перевірка на академічний плагіат	04.12.21	
8.	Перевірка завідувачем кафедри, рецензування	05.12.21–07.12.21	

Дата видачі завдання *8.11.2021 р.*

Студент _____
(підпис)

Керівник роботи _____ *проф. Колендовська М.М.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи має: 67 с., 39 рис., 2 табл., 5 додатків, 47 джерел.

МОБІЛЬНІ РОБОТИ, ШТУЧНИЙ ІНТЕЛЕКТ, КОМП’ЮТЕРНИЙ ЗІР ІГРОВИЙ ДВИГУН, ПЕРЕШКОДИ

Об'єкт дослідження – мобільні роботи.

Предмет дослідження – системи комп’ютерного зору мобільних роботів.

Мета кваліфікаційної роботи роботи – розробити алгоритм подолання перешкод мобільним роботом.

Методи дослідження – теоретичний аналіз, комп’ютерний зір, математичне моделювання, обробка даних.

У даній роботі наведено класифікацію мобільних роботів, види комп’ютерного зору, детальний аналіз систем комп'ютерного зору, огляд методів розпізнавання об'єктів, розробка алгоритму і моделі поведінки мобільних роботів, розроблено алгоритм задачі оцінки перешкоди, яка вирішена за допомогою навчання робота комп’ютерному зору.

ABSTRACT

The explanatory note of the qualification work has: 67 pages, 39 figures, 2 tables, 5 appendices, 47 sources.

MOBILE WORKS, ARTIFICIAL INTELLIGENCE, COMPUTER VISION GAME ENGINE, INTERFERENCES

The object of study - mobile work.

The subject of research - is computer vision systems of mobile robots.

The purpose of the qualification work - is to develop an algorithm for overcoming obstacles with a mobile robot.

Research methods - theoretical analysis, computer vision, mathematical modeling, data processing.

This paper presents the classification of mobile robots, types of computer vision, detailed analysis of computer vision systems, review of object recognition methods, development of algorithms and models of behavior of mobile robots, developed an algorithm for obstacle assessment, which is solved by training robots computer vision.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

МР – мобільний робот;

ШІ – штучний інтелект;

ІД – ігровий двигун;

БПЛА - безпілотний літальний аппарат;

КЗ – комп’ютерний зір;

ПЗ – прямий в’язок;

ЗЗ – зворотній зв’язок;

ЛН – лазерна навігація;

UE – unreal engine;

DLL – dynamic-link library;

НС – навігаційна система;

ЗД – зовнішні датчики.

ЗМІСТ

ВСТУП

У сучасному світі використання мобільних роботів звична справа в абсолютно різних сферах нашого життя. Діапазон застосування неймовірно широкий:

- повна автоматизація багатьох процесів зводить нанівець участь людей у виробництві;
- мобільні роботи використовуються для дослідження космічного простору і океанських глибин;
- проведення найскладніших хірургічних операції,
- використання в виробництві протезних кінцівок і деяких внутрішніх органів;
- застосування у військовій галузі (безпілотники, дрони);
- суспільна безпека: в арсеналі спецслужб і поліцейських підрозділів;
- сфера послуг: доставка продуктів, медикаментів і товарів різного призначення, що надзвичайно важливо і актуально на сьогоднішній день в ситуації, що склалася з пандемією;
- побутові умови: прибирання будинків роботами-пилососами, автопілот автомобілів і багато іншого.

Мобільні роботи виділяються не великими розмірами і можливістю пересування, економічністю утримання, порівняно зі стаціонарними роботами. Проте для гарної роботи робота потрібно навчити бачити, щоб він мав змогу долати перешкоди на своєму шляху. Саме тому, актуальність створення алгоритму проходження перешкод для мобільних роботів так важлива на даний момент.

1 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ МОДЕЛЕЙ ПОВЕДІНКИ МОБІЛЬНИХ РОБОТІВ ПРИ РОЗПІЗНАВАННІ ПЕРЕШКОД

1.1 Аналіз розвитку існуючих роботів

Робот – це автоматичний пристрій, призначений для здійснення різноманітних механічних операцій, що діє за заздалегідь закладеною програмою [1].

Робот зазвичай отримує інформацію про стан навколишнього середовища за допомогою датчиків (технічних аналогів органів чуття живих організмів). Робот може самостійно здійснювати виробничі та інші операції, частково чи повністю замінюючи працю людини. При цьому робот може мати зв'язок з оператором, отримуючи від нього команди (ручне управління), так і діяти автономно, відповідно до закладеної програми (автоматичне управління) [1].

Призначення роботів можуть бути найрізноманітнішими, від розважальних та прикладних і до суто виробничих. Зовнішній вигляд роботів різноманітний за формою і змістом, може бути яким завгодно, хоча нерідко в конструкціях вузлів запозичують елементи анатомії різних живих істот, придатні для завдання.

В інформаційних технологіях «роботами» також називають деякі автономні програми, наприклад, боти або пошукові роботи [1-3].

Робототехнікою, серйозно почали займатися з кінця 1990-х. Японія позначила робототехніку в якості одного з семи національних пріоритетів, поставивши за мету стати світовим лідером у галузі домашньої робототехніки [1-5].

Китай не оголошував офіційно своїх планів у цій галузі, однак, судячи з окремих повідомлень про конкретні проекти [1-10], веде роботи у галузі військової, планетарної та інших видів робототехніки. Південна Корея

здійснює план розвитку робототехніки [1-11], одним із головних завдань якого є стратегічне фокусування на потребах окремих ринків, наприклад, роботизовані системи спостереження для Близького Сходу, медичні роботи для США та ЄС, а також роботи-прибиральники, призначені для Китаю та Південно-Східної Азії [1-13].

1.2 Аналіз існуючих типів роботів

Відштовкуючись віж літературних джерел [5-25] можна зробити висновок, що існує безліч різних роботів, які виконують різні завдання для яких були створені та запрограмовані. Проаналізувавши велику кількість роботів можна виділити основні їх типи:

Побутова робототехніка. Створення домашніх роботів [5-26].

Медична робототехніка. Створення медичних роботів. Їх впровадження дозволить, зокрема, вирішити проблему поширення вірусів та інфекцій самими лікарями та запобігнути захворюванню медпрацівників [5-27].

Персональна робототехніка. Розробка персональних роботів невеликих, недорогих, простих та зручних у використанні. Спеціальний вібротактильний костюм, за допомогою якого можна навчити людину будь-яким руховим навичкам або прискорити одужання пацієнтів, які проходять реабілітацію після різних неврологічних травм або універсальний особистий помічник-гуманоїд [5-28].

Планетарна робототехніка. Проектування роботів для дослідження планет [5-29].

Військова робототехніка, що займається створенням наступних видів [6]:

БПЛА - безпілотний літальний апарат. Серед військових БПЛА можуть бути виділені: тактичні, малі, малі тактичні та надмалі [6-1];

НМР наземний мобільний робот. Автоматично керований (роботизований) наземний транспортний засіб; серед військових НМР

розрізняють: тактичні та малі, а також роботизовані транспортні засоби для евакуації поранених [6-2];

Морські роботи. Автоматично керований (роботизований) морський транспортний засіб; роботи цього класу (в основному військового призначення) поділяються на надводні та підводні [6-3].

Телеробототехніка. Створення телероботів (роботів, що дистанційно керуються телеоператором) [7]. Роботи для МНС, МО тощо;

Промислова робототехніка. Розробка промислових роботів, число видів яких перевищує три десятки [8]. Лідером тут є Японія вона має парк, що налічує понад 350 тис. індустріальних роботів. Загалом у світі налічується 1,1 млн. індустріальних роботів та близько 17 млн. роботів інших видів [9-3].

Еволюційна робототехніка. Вивчення методів еволюційних обчислень для розробки штучних нервових систем роботів [10].

Польова робототехніка. Дослідження та створення автономних рухомих роботів для виконання тих чи інших робіт у природних, іноді екстремальних умовах [11].

Біометрична робототехніка. Дослідження та створення роботів з біометричними можливостями, наприклад, з реакцією на дотик [12].

Біологічна робототехніка. Дослідження та проектування біологічних роботів (біороботів, або біоботів); повністю біологічні роботи не мають у своїй основі кремнієвих компонентів, являють собою штучний інтелект на базі органічної субстанції, здатні зростати за рахунок появи нових мікроорганізмів, що розмножуються під впливом світла, тепла та поживних речовин, можуть вирішувати деякі обчислювальні та логічні завдання [13]. У перспективі можливе створення складніших біороботів, здатних самоорганізовуватися, працювати у військовій, виробничій та медичній сферах [13-2].

Мікроробототехніка. Розробка надмініатюрних робототехнічних пристроїв [14].

Наноробототехніка. Створення нанороботів пристроїв розміром одиниці і десятки нанометрів, які зможуть самостійно маніпулювати

окремими атомами речовини. Наноробототехніка входить до науки науки про наномір, нанонауки [15-3].

Нейроробототехніка. Міждисциплінарний напрямок на стику штучного інтелекту, біомеханіки, неврології, робототехніки, біо- і психофізики, мета дослідження проблем зв'язку між центральною нервовою системою і м'язової активністю людини, розробка біонічних інтерфейсів, створення штучних частин тіла (в організм замість втрачених та управління ними, створення допоміжних пристроїв (наприклад, екзоскелетів)) для реабілітації після травм та розширення фізичних можливостей людини [16].

Транспортний робот. Автоматична машина, що представляє собою сукупність маніпулятора, пристрої керування, що перепрограмується, і ходового пристрою [17].

Аптечний робот (робот для аптеки). Міні-складське обладнання, яке встановлюється в аптеці, аптечних складах та медичних закладах для оптимізації зберігання, пошуку та видачі медикаментів до робочого місця фармацевта (першого столу) або для продажу безпосередньо покупцю [18].

Проаналізувавши стан та використання сучасних роботів [19-4], видно, що роботи зайняли своє місце в різноманітних областях:

Роботи кур'єри. Проект автоконцерну Ford та Agility Robotics, двуногий механічний кур'єр Digit, подорожує безпілотною вантажівкою. Після прибуття робот самостійно розкладається, забирає посилку з машини та несе до порога адресата [20].

Digit оснащений лідарами та камерами, вміє підніматися сходами, розпізнавати перешкоди, утримувати рівновагу після зіткнення з об'єктами [20].

Роботи, що вчаться. Робот Atlas створений компанією Boston Dynamic спритно застрибує на різні поверхні, робить сальто назад, гарно балансує та тримає рівновагу [21].

Нещодавно Boston Dynamic представила Atlas з покращеними алгоритмами координації рухів та поведінки у просторі. Інформація, зібрана з

датчиків, Atlas використовує для побудови моделі зовнішнього середовища та виявлення поверхонь, придатних для постановки ноги [21-2].

Роботи охоронці. Cobalt Robotics вивела на комерційний ринок роботів-охоронців-помічників. Пристрій нагадує кулер для води, він перевіряє офіси, склади, паркування [22].

Роботи оснащені датчиками, камерами, тепловізорами, покладаються на штучний інтелект та машинне навчання, здатні виявляти протікання води, відчинені двері, прояви агресії [22-2]. Про порушення Cobalt повідомляє центральний пункт моніторингу.

Роботи помічники. Роботи орієнтуються у просторі, уникають зіткнень один з одним, здатні переносити вантаж масою до 500 кілограм [23].

Роботи художники. Дрони намалювали картину розміром 14 на 12 метрів. Безпілотниками керувала єдина центральна система [24]. Камери відстежували рух апаратів, а програма направляла за потрібним маршрутом.

Віртуальні роботи (боти). Крім звичного розуміння значення «робот» та асоціації, що відразу нами згадується так само є й інше – віртуальне або краще сказати бот [25].

Бот – програма-робот, що керується комп'ютером (штучним інтелектом) та імітує партнерів і противників у грах, мережевих поєдинках, командних битвах тощо. Програма-бот заснована на модулі штучного інтелекту, який адаптований до особливостей цієї гри: жанру, ролі персонажа, конкретної карти.

Боти зазвичай написані на C/C++ як самостійний незалежний додаток або плаґін чи просто клієнтська бібліотека (англ. Dynamic-link library, DLL) для конкретного ігрового двигуна [26].

1.3 Аналіз алгоритмів пересування та прийняття рішень мобільних роботів

Розглянемо різновиди алгоритмів пересування та прийняття рішень роботами.

1.3.1 Аналіз алгоритму управління без зворотного зв'язку

У завданнях управління зазвичай існують два об'єкти: керуючий та керований [27]. У найпростішому варіанті від керуючого об'єкта надходить команда і керований виконує її, нічого не повідомляючи про результат або про умови роботи, що змінилися. Суть прямого зв'язку представлена схемою на рис. 1.1.

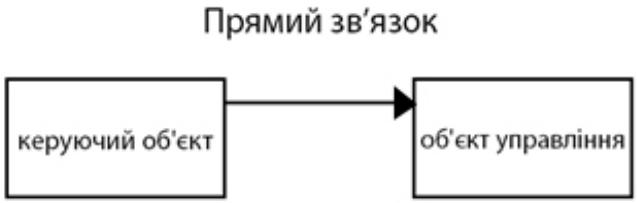


Рисунок 1.1 – Прямий зв'язок в управлінні

З погляду мобільного робота керуючий об'єкт – це його контролер із запущеною програмою, об'єкт управління – це його колеса та корпус (шасі). Керуючі команди контролер подає на мотори, при прямому зв'язку керуючись показаннями своїх внутрішніх годинників - таймера.

Алгоритми руху протягом заданого часу вперед та назад. Для руху вперед використовуються команди керування двигунами. Ці команди просто включають двигуни. Особливість NXT полягає в тому, що після закінчення виконання програми зберігаються всі установки в поведінці робота, але на двигуни перестає подаватися напруга. Таким чином, проходить пуск і відразу плавне гальмування як видно з рис. 1.2 [27].

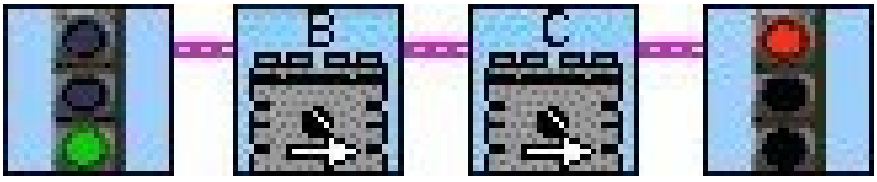


Рисунок 1.2 – Схема ввімкнення моторів [27]

Нижче наведено приклад програми схеми ввімкнення моторів:

```
Task main()
{
motor[motorB] = 100; // мотори вперед із
motor[motorC] = 100; // максимальна потужність
}
```

Обидві команди виконуються майже миттєво. Якщо відразу слідом за ними вимкнути мотори, то візок просто смикнеться і залишиться стояти на місці, як видно з рис. 1.3 [27].

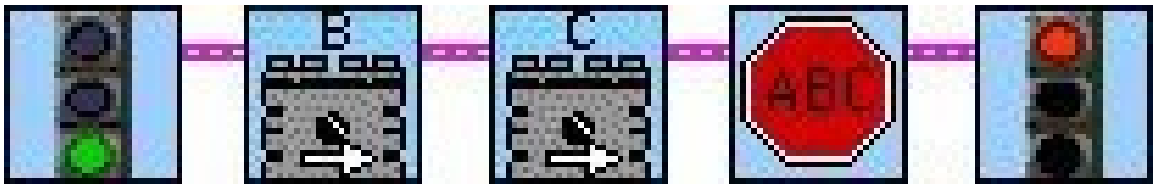


Рисунок 1.3 - Зупинка при спробі розпочати рух [27]

Нижче наведено приклад програми зупинки при спробі розпочати рух:

```
Task main() { motor[motorB] = 100; motor[motorC] =
100; motor[motorB] = 0; // стоп мотор motor[motorC] =
0; }
```

Таким чином, для здійснення руху потрібна деяка затримка перед вимкненням двигунів. Команди очікування не роблять жодних конкретних дій, зате дають можливість двигунам виконати свою частину роботи як видно з рис. 1.4 [27].

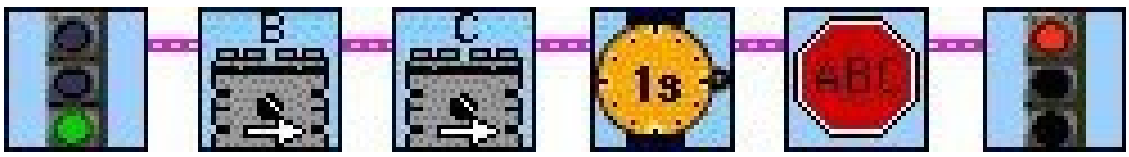


Рисунок 1.4 - Правильний порядок керування моторами [27]

Нижче наведено приклад програми з правильним порядком керування моторами:

```
Task main() { motor[motorB] = 100; motor[motorC] = 100; wait1Msec(1000); // Чекати 1000 мс motor[motorB] = 0; motor[motorC] = 0; }
```

Рух уперед чи назад, визначається напрямом обертання двигунів. Для зміни напряму не потрібна зупинка як видно з рис. 1.5 [27]:

Рисунок 1.5 - Проїхати секунду вперед, секунду назад та зупинитися [27]

Нижче наведено приклад програми руху на одну секунду вперед, назад та зупинитися:

```
Task main() { motor[motorB] = 100; motor[motorC] = 100; wait1Msec(1000); motor[motorB] = -100; // "Повний назад" motor[motorC] = -100; wait1Msec(1000); motor[motorB] = 0; motor[motorC] = 0; }
```

У момент зміни напрямку на високій швидкості можливе занесення. Плавне гальмування можливе. Для цього перед подачею команди "назад" з моторів знімається напруга і робот деякий час їде за інерцією як видно з рис. 1.6 [27].

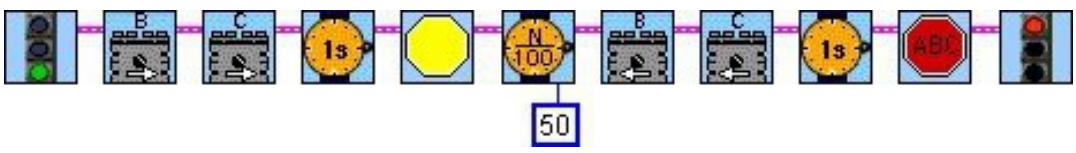


Рисунок 1.6 - Перед зміною напряму півсекунди їхати за інерцією [27]

Нижче наведено приклад програми перед зміною напряму півсекунди їхати за інерцією:

```
Task main() { motor[motorB] = 100; motor[motorC] = 100; wait1Msec(1000); // Включити плаваючий режим керування двигунами bFloatDuringInactiveMotorPWM = true; motor[motorB] = 0; motor[motorC] = 0; wait1Msec(500); motor[motorB] = -100; motor[motorC] = -100; wait1Msec(1000); // Включити режим гальмування bFloatDuringInactiveMotorPWM = false; motor[motorB] = 0; motor[motorC] = 0; }
```

Алгоритм руху на поворотах. Для виконання повороту дома достатньо включити мотори в різні боки. Тоді робот обертатиметься приблизно навколо центру осі провідних коліс зі зміщенням у бік центру тяжіння. Для точного повороту треба підбирати час у сотих частках секунди. Однак при зміні заряду батарейок доведеться вводити нові параметри повороту як видно з рис. 1.7 [27]:

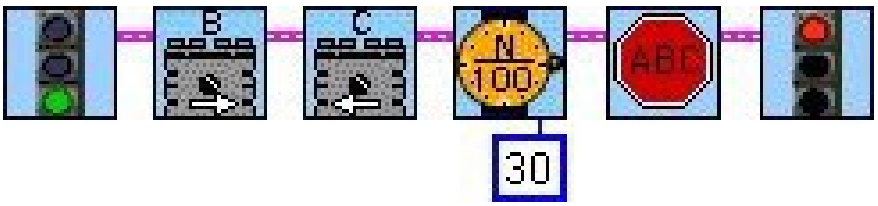


Рисунок 1.7 – Поворот на місці [27]

Нижче наведено приклад програми повороту на місці:

```
Task main() { motor[motorB] = 100; // Мотори у різні motor[motorC] = -100; // Сторони wait1Msec(300); motor[motorB] = 0; motor[motorC] = 0; }
```

Існує інший тип поворотів. Якщо один з моторів зупинити, а інший включити, то обертання відбуватиметься навколо мотора, що стоїть. Поворот вийде більш плавним. Приклад наведено на рис. 1.8 [27].

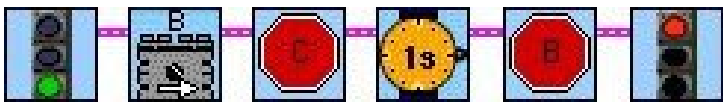


Рисунок 1.8 - Плавний поворот [27]

Нижче наведено приклад програми плавного повороту:

```
Task main() { motor[motorB] = 100; motor[motorC] = 0; wait1Msec(1000); // Повертається тільки мотор B motor[motorB] = 0; }
```

Алгоритм руху по квадрату. Використовуючи отримані знання управління моторами, можна запрограмувати рух квадратом або іншим багатокутником за допомогою циклу або безумовного переходу як видно з рис. 1.9 [27-7].

Рисунок 1.9 - Рух багатокутником з плавними поворотами [27]

Нижче наведено приклад програми руху багатокутником з плавними поворотами:

```
Task main()
{while (true){ motor[motorB] = 100; motor[motorC] = 100; wait1Msec(1000); motor[motorC] = 0; wait1Msec(1000); motor[motorB] = 0;
}
```

Уточнивши тривалість поворотів та кількість повторень, навчимо візок об'їжджати квадрат по периметру 1 раз. Для точності поворотів знизимо потужність двигунів приблизно вдвічі як видно з рис. 1.10 [27].

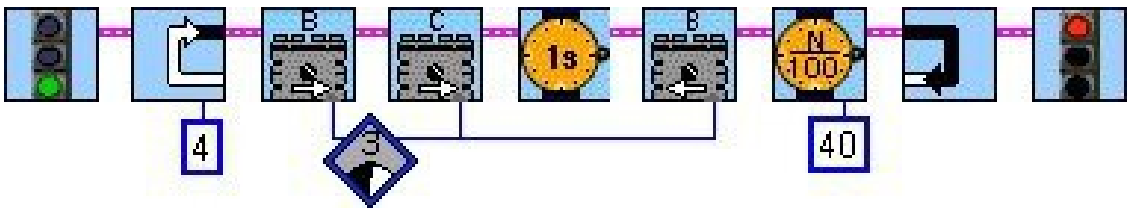


Рисунок 1.10 - Для повороту на 90 градусів [27]

Нижче наведено приклад програми для повороту на 90 градусів:

```
Task main() { for(int i=0;i<4;i++){ // Цикл виконується 4 рази motor[motorB] = 50; motor[motorC] = 50; wait1Msec(1000); motor[motorC] = -50; wait1Msec(400); motor[motorB] = 0; } }
```

1.3.2 Аналіз алгоритмів управління зі зворотним зв'язком

Зворотній зв'язок. Поява зворотного зв'язку в системі означає те, що об'єкт, що управляє, починає отримувати інформацію про об'єкт управління як видно з рис. 1.11.

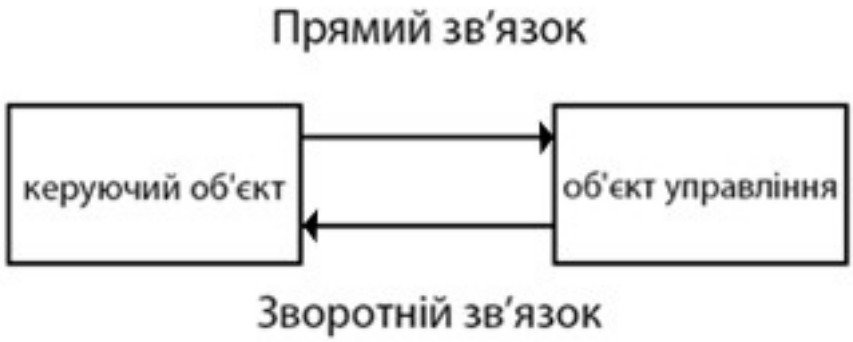


Рисунок 1.11 - Управління із зворотним зв'язком

Зворотний зв'язок здійснюється за допомогою датчиків, прикріплених, наприклад, корпусу робота [28]. Дані надходять до контролера, який є керуючим об'єктом.

Алгоритм руху «Точні переміщення». Щоб поворот не залежав від заряду батарейок, можна скористатися вбудованим у двигуни датчиком обертів, «енкодером», який дозволяє робити вимірювання з точністю до 1 градуса. Для більш ефективного управління задіяні «просунуті» команди, вважаючи, що при повороті візка на 90 градусів ліве колесо повертається на 250 градусів навколо своєї осі як видно з рис. 1.12 [27].

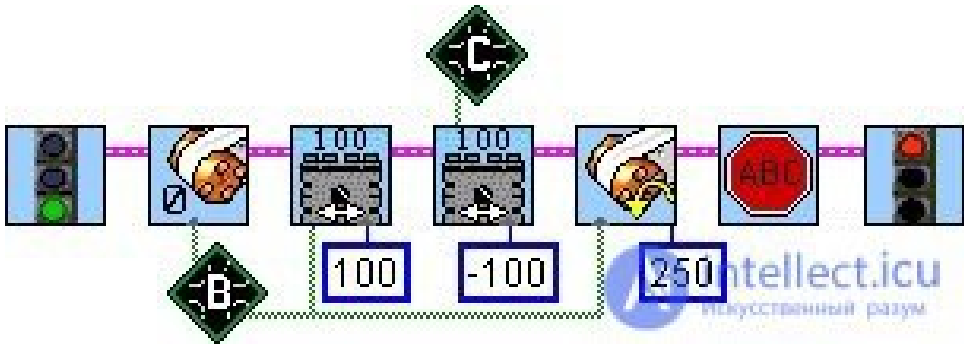


Рисунок 1.12 - Точний поворот на місці [27]

Нижче наведено приклад програми точного повороту на місці:

```
Task main() { nMotorEncoder[motorB]=0; // Ініціалізація енкодера motor[motorB] = 100; motor[motorC] = -100; // Порожній цикл очікування показань енкодера while(nMotorEncoder[motorB]<250); motor[motorB] = 0; motor[motorC] = 0; }
```

Алгоритм руху вздовж лінії. Для цього потрібне чорне коло на білому фоні. Датчик освітленості слід висунути трохи вперед, щоб він утворював разом із провідними колесами рівносторонній або хоча б рівнобедрений прямокутний трикутник як видно з рис. 1.13 [27].

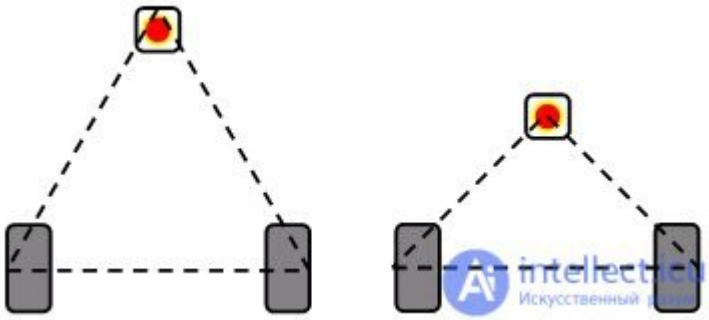


Рисунок 1.13 - Варіанти розташування датчика освітленості щодо провідних коліс [27]

Завдання робота таке: рухатися вздовж кола по межі чорного та білого. Застосуємо релейний (чи пропорційний) регулятор. Тільки алгоритм записаний не у вигляді розгалуження, а з використанням блоків «Чекай темніше» та «Чекай світліше». Без модифікаторів передбачається, що датчик освітленості підключений до першого порту, а мотори подається максимальна потужність як видно з рис. 1.14 [27].

Рисунок 1.14 - Алгоритм руху по лінії з одним датчиком освітлення [27]

Перед стартом ставимо робота на лінію так, щоб датчик був трохи ліворуч. За алгоритмом робот плавно повертає праворуч, поки освітленість не знизиться на 5 пунктів (за замовчуванням). Потім повертає ліворуч, доки освітленість не підвищиться на 5 пунктів. Рух виходить схожим на «змійку».

Можливі проблеми:

1. Робот крутиться на місці, не заїжджаючи на лінію. У цьому випадку слід або стартувати з іншого боку лінії, або змінити підключення двигунів до контролера місцями;
2. Робот проскакує лінію, не встигаючи зреагувати. Слід знизити потужність двигунів;

3. Робот реагує на дрібні перешкоди на білому, не доїжджаючи до чорного. Потрібно збільшити поріг чутливості датчика (наприклад, не на 5, а на 8 пунктів). Взагалі, це число можна розрахувати. Для цього слід зняти показання датчика на білому, потім на чорному відняти одне з іншого і поділити навпіл. Наприклад, (56 - 40) / 2 = 8.

Удосконалену програму показано на рис. 1.15 [27]

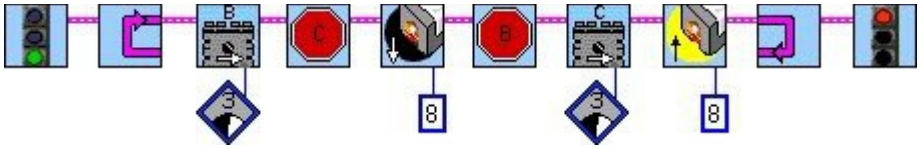


Рисунок 1.15 - Алгоритм руху по лінії з одним датчиком освітленості: знижено швидкість, збільшено різницю між чорним і білим [27]

Більш стійко алгоритм працює, якщо використовувати мотори з керуванням швидкістю -100 ... 100. У цьому випадку є можливість відрегулювати плавність повороту відповідно до кривизної лінії як видно з рис. 1.16 [28-6].

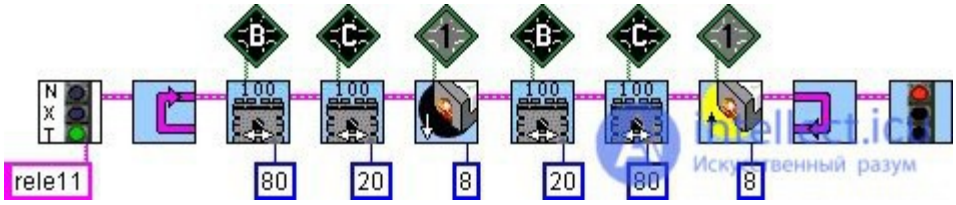


Рисунок 1.16 - Алгоритм руху по лінії з одним датчиком освітленості: покращено керування моторами [27]

У цьому алгоритмі мотори, що пригальмовують, на повороті не зупиняються повністю, а лише знижують швидкість до 20 пунктів. Це робить поворот плавнішим, але може призвести і до втрати лінії на різкому повороті.

П-регулятор. Проаналізуємо для порівняння, як працюватиме П-регулятор для одного датчика як видно на рис. 1.17 [27].

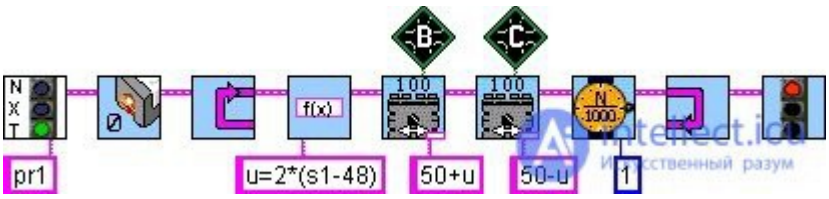


Рисунок 1.17 - Алгоритм руху лінії з одним датчиком освітленості на пропорційному регуляторі [27]

Число 48, використане у формулі управління u, - це середнє арифметичне показань датчика освітленості на чорному і білому, наприклад (40 + 56) / 2 = 48. Однак показання датчиків часто змінюються з різних причин: інша поверхня, зміна загальної освітленості в приміщенні, невелика модифікація конструкції та ін. Тому має сенс навчити робота самостійно обчислювати середнє арифметичне, тобто значення межі білого та чорного.

Є кілька способів виконати калібрування датчика. У найпростішому випадку замість обчислення середнього арифметичного знижується значення білого. Сенс способу в тому, що робот знімає свідчення на білому, віднімає з нього деяке передбачуване значення і число вважає межею білого і чорного. Наприклад, 56 – 7 = 49 вважатимуться значенням сірого як видно з рис. 1.18 [27].

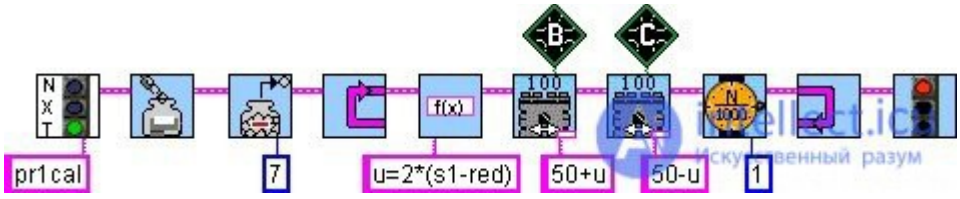


Рисунок 1.18 - Алгоритм руху по лінії з одним датчиком освітленості на пропорційному регуляторі з попереднім калібруванням (визначенням сірого значення) [27]

Нижче наведено приклад програми алгоритму руху по лінії з одним датчиком освітленості на пропорційному регуляторі з попереднім калібруванням (визначенням сірого значення):

```
task main() { int u, v=50; float k=2; int red=SensorValue[S1]-7; while(true) { u=k*(SensorValue[s1]-red); motor[motorB]=v+u; motor[motorC]=v-u; wait1Msec(1); } }
```

За замовчуванням значення освітленості з датчика на порту 1 зчитується червоний контейнер, після чого воно зменшується на число 7, і у формулі управління u використовується вже змінене значення червоного контейнера red. Якщо вказувати всі модифікатори, програма виглядатиме так, як показано на рис. 1.19 [27].

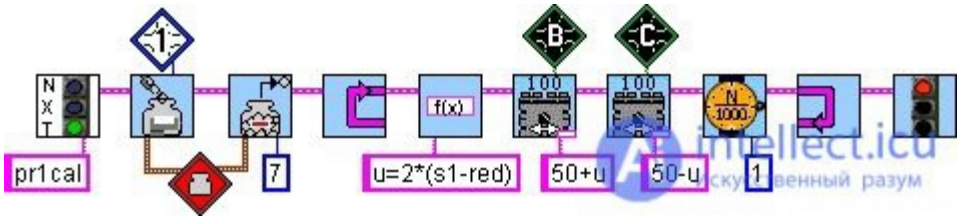


Рисунок 1.19 - Алгоритм руху по лінії з одним датчиком освітленості на пропорційному регуляторі – з модифікаторами [27]

Треба мати на увазі, що такий спосіб калібрування не враховує всіх можливих варіантів, а лише заощаджує час на програмування та налагодження. Якщо ж часу достатньо, є інший спосіб, при якому справді проводиться розрахунок середнього арифметичного показань датчика освітленості на чорному та білому як видно з рис. 1.20 [27].

Рисунок 1.20 - Алгоритм руху по лінії з одним датчиком освітленості на пропорційному регуляторі з розрахунком сірого значення [27]

Нижче наведено приклад програми алгоритму руху по лінії з одним датчиком освітленості на пропорційному регуляторі з розрахунком сірого значення:

```
task main() { int u, v=50; float k=2; int c4=SensorValue[S1]; PlaySound(soundBeepBeep); wait1Msec(2000); int c5=SensorValue[S1]; PlaySound(soundBeepBeep); int grey=(c4+c5)/2; while(true) { u=k*(SensorValue[S1]-grey); motor[motorB]=v+u; motor[motorC]=v-u; wait1Msec(1); } }
```

Алгоритм має деяку незручність: при запуску потрібно бути уважним і не пропустити звукового сигналу, після якого робота треба перемістити так, щоб датчик освітленості виявився над білим полем. Спочатку слід помістити робота точно над чорною лінією. У контейнері з номером 4 (позначається c4) буде збережено значення чорного, у контейнері з номером 5 (c5) – значення білого. У змінну grey міститься значення сірого, що використовується в регуляторі. Відразу після другого звукового сигналу робот розпочне рух.

Калібрування можна зробити більш керованим. Для цього після кожного зчитування даних необхідно вставити очікування будь-якої зовнішньої події, наприклад, натискання на датчик торкання, зменшення відстані на ультразвуковому датчику або просто натискання на кнопку NXT.

Розглянемо приклад із додатковим датчиком торкання, приєднаним до другого порту. Запустити програму має сенс, акуратно встановивши візок датчиком освітлення над чорною лінією як видно з рис. 1.21 [27].

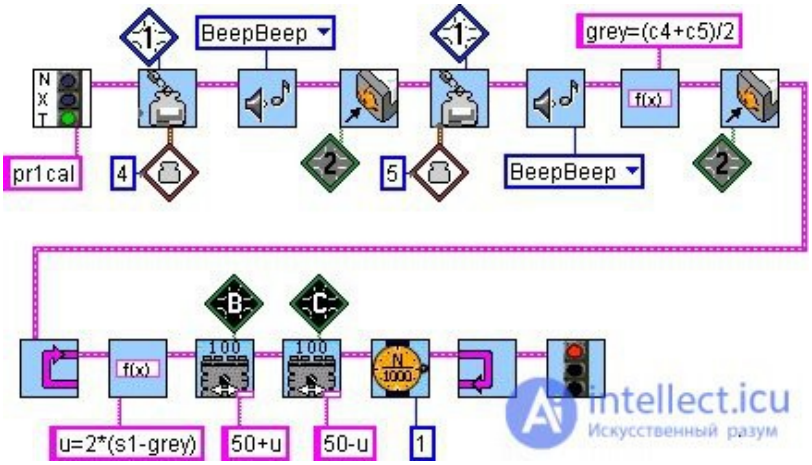


Рисунок 1.21 - Калібрування датчика освітленості з очікуванням торкання [27]

Нижче наведено приклад програми калібрування датчика освітленості з очікуванням торкання:

```
task main() { int u, v=50; float k=2; int c4=SensorValue[S1]; PlaySound(soundBeepBeep); while(SensorValue[S2]==0); // Чекай, доки натиснуто wait1Msec(100); // Захист від залипань while(SensorValue[S2]==1); // Чекай, поки натиснуто wait1Msec(100); int c5=SensorValue[S1]; PlaySound(soundBeepBeep); int grey=(c4+c5)/2; while(SensorValue[S2]==0); wait1Msec(100); while(SensorValue[S2]==1); while(true) { u=k*(SensorValue[S1]-grey); motor[motorB]=v+u; motor[motorC]=v-u; wait1Msec(1); } }
```

Після першого звукового сигналу потрібно переставити візок так, щоб датчик освітленості виявився над білим. Після другого сигналу підготуватися до старту (датчик освітленості на межі між чорним та білим) і натисканням кнопки стартувати.

Аналогічний досвід можна провести за допомогою датчика відстані замість датчика натискання. Перевага тут у тому, що старт робота здійснюватиметься безконтактно. Це допоможе стартувати у точно

вибраному положенні. Тільки треба бути уважним та несвоєчасно не провести рукою біля датчика відстані як видно з рис. 1.22 [27].

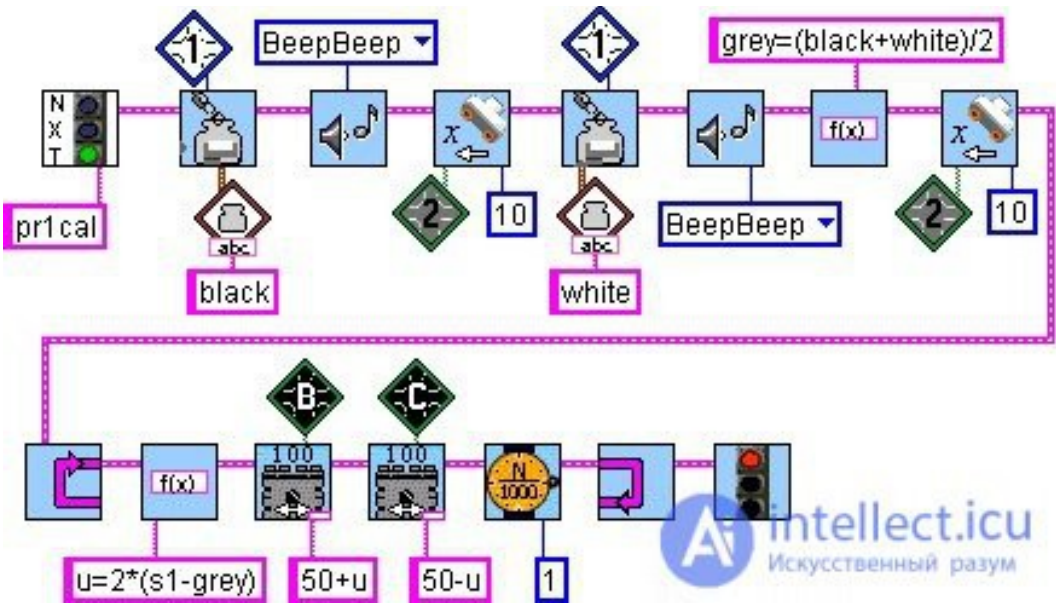


Рисунок 1.22 - Калібрування датчика освітлення з очікуванням об'єкта (руки) [27]

У прикладі використані іменовані контейнери (black і white), які насправді є змінними, як у звичайній мові програмування. Звуковий сигнал між двома очікуваннями зміни відстані сприяє тому, щоб робот не зреагував двічі поспіль на одне наближення руки.

1.4 Аналіз навігаційних систем та систем датчиків мобільних роботів

Від якості та типу системи навігації для роботизованого засобу залежить, наскільки швидко робот орієнтуватиметься у просторі та як добре виконуватиме свої обов'язки.

Система датчиків поділяється на такі типи [29]:

1. Зовнішні датчики;
2. Лазерна навігація;
3. Відеоспостереження;
4. Спеціалізовані датчики.

1.4.1 Зовнішні датчики

Зовнішні датчики потрібні, щоби сканувати простір, уникати перешкод і вибудовувати оптимальний маршрут для пересування [29-2]. На таб. 1.1 можна побачити які типи датчиків та для чого потрібні:

Таблиця 1.1 – Зовнішні датчики

Тип датчика	Навіщо потрібен
Контактний	Для обходу перешкод під час зіткнення.
Безконтактний	Реєструє перешкоду та дозволяє зупинитися до зіткнення.
Ультразвуковий далекомір	Працює за принципом ехолокації та виключає можливість удару об стіни

Зовнішні датчики дозволяють зберегти робота в цілісності та вберегти його від зіткнень із меблями чи стінами.

1.4.2 Лазерна навігація

Найбільш просунутий тип орієнтування в просторі передбачає можливість побудови карти приміщення [30]. Спеціальні лазерні далекоміри фіксують відстань до стін, меблів та зберігають дані в пам'яті пристрою.

Ще вони допомагають вибудовувати віртуальні стіни. Віртуальна стіна - це спосіб поділу простору на зони, де потрібно виконувати особливі роботи, наприклад, прибирання або фарбування та протилежні зони, де немає необхідності відвідувати роботи [30-6].

1.4.3 Відеокамери

Камера встановлюється у верхній, найвищій точці панелі та сканує інформацію зі стін, підлоги, стелі та меблів. Такий прилад рухається по

прямих лініях і по черзі переходить від кімнати до кімнати. При цьому можуть використовуватися алгоритми комп'ютерного зору та розпізнавання образів [31].

1.4.4 Захист від застрягань

Не всі предмети по дорозі потрапляють у поле зору робота. Наприклад, якщо перешкода досить низька, то ультразвуковий датчик може не помітити. Або покрита тканинною оббивкою поверхня зовсім поглинає ультразвукові сигнали, тобто не відбиває їх на чутливий елемент [32].

Не побачивши перешкоду (тапок або ніжку стільця), робот може застрягти і буде нескінченно намагатися продовжувати рух уперед. Однак в кімнаті рух не повинен бути нескінченним. Якщо, від однієї стінки до іншої робот може дістатися за 10 с., та за цей час він не побачить жодної перешкоди, можна з упевненістю стверджувати, що сталося застрявання і треба вжити екстрених заходів. При такій ситуації слід просто від'їхати назад та розвернутися. Допоможе цьому «сторожовий таймер». Такі пристрої застосовуються у мікроконтролерах та захищають їх від зависань як видно з рис. 1.23 [27].

Рисунок 1.23 - Якщо на сторожовому таймері набереться 10 секунд, включається захист від застрягань [27]

Можна помітити в програмі групи блоків, що повторюються. Їх варто об'єднати у підпрограму, яка здійснюватиме від'їзд назад із розворотом. Таким чином, підпрограма від'їзду викликатиметься у двох випадках: 1) за наявності перешкоди; 2) при спрацьовуванні сторожового таймера. Як видно з рис. 1.24 [27].

Рисунок 1.24 – Захист від застрягань із використанням підпрограм [27]

Поворот за кут. Необхідно обмежити реакцію робота на «нескінченність». Коли в полі видимості немає об'єкта, показання датчика відстані NXT дорівнюють 250 або 255 см. Якщо це число потрапляє на пропорційний регулятор, робот починає обертатися на місці. А у ситуації, коли роботу слід завернути за кут, саме це й станеться.

Для об'їзду предметів потрібно ввести контроль показань датчика відстані: при різкій зміні робот повинен робити висновок про можливий поворот, який треба буде робити з іншими коефіцієнтами або просто з постійним значенням впливу, що управляє.

Розглянемо приклад повороту праворуч «за кут». Якщо робот рухається з відривом L від стіни, те й поворот, очевидно, він виконуватиме по колу з радіусом L як видно з рис. 1.25 [27].

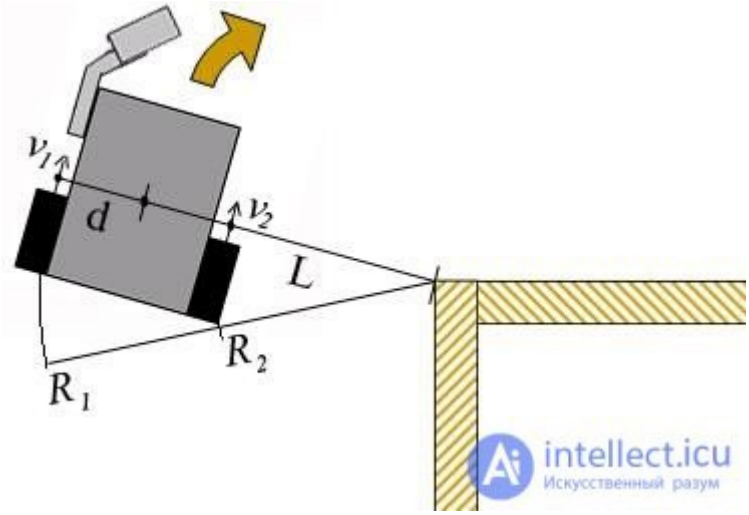


Рисунок 1.25 - Виконує поворот при втраті контакту зі стінкою [27]

Розрахувавши, яким має бути відношення швидкостей коліс, щоб радіус повороту дорівнював L. Для цього достатньо виміряти відстань між передніми колесами. Нехай у нашому роботі воно дорівнюватиме k = 16 см, а його половина d = 16 / 2 = 8 см. Тоді ліве і праве колеса рухаються по колам радіусів, відповідно, R1 = L + d і R2 = L – d. Пройдені ними шляхи за одиницю часу повинні бути пропорційні радіусам, отже швидкості точок кріплення коліс v1 і v2 пов'язані наступним відношенням 1.1:

$$v1 = R1.\ v2\ R2 \qquad (1.1)$$

Виражаючи швидкість переміщення коліс через базову швидкість v і невідому x, а радіуси через L, отримуємо наступне відношення 1.2:

$$v + x\ L+d = ,\ vL+xL-vd-xd = vL+vd-xL-xd\ ,\ 2xL=2vd\ ,\ v - x\ Ld\ x = vd,\ v1 = v + vd = v(1+ d),\ v2 = v - vd = v\ (1-d) \qquad (1.2)$$

Лінійна швидкість v пропорційна кутової швидкості колеса ω, яка у свою чергу пропорційна потужності, що подається на мотори (в режимі гальмування). Преведено закон управління до стандартного вигляду, що дозволяє задати керуючий вплив на час повороту за кут. Таким чином, отримуємо розрахунок для керування двигунами робота 1.3:

$$u=v*8/L;\ motor[motorB]=v+u;\ motor[motorC]=v-u \qquad (1.3)$$

Коли відстань до стіни стає більше 2L (використовується такий поріг видимості), відкривається поворот за кут, вплив, що управляє, починає обчислюватися за наведеними формулами як видно з рис. 1.26 [27].

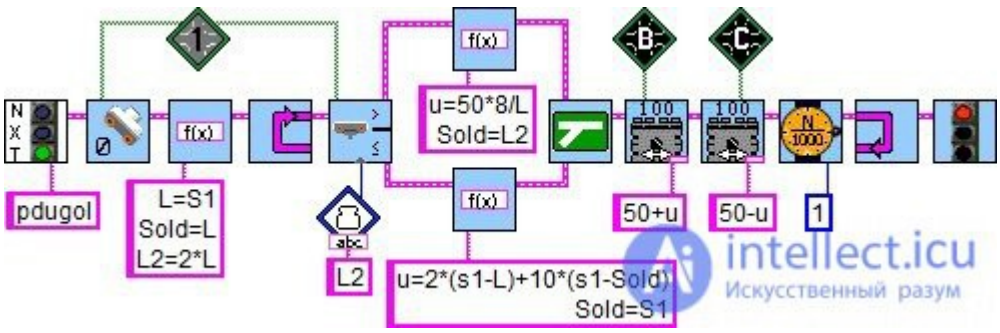


Рисунок 1.26 - Об'їзд предметів на заданій відстані за правилом правої руки [27]

Нижче наведено приклад програми об'їзд предметів на заданій відстані за правилом правої руки:

```
task main() { float u, k1=2, k2=10; int v = 50, d = 8, Sold, L, Snew; Sold=L=SensorValue[S1]; // Запам'ятали початковий стан
while (true) { Snew = SensorValue [S1]; // Отримали показання датчика
if (Snew>L*2) { u=v*d/L; Sold = L * 2; } else {u = k1 * (Snew-L) + k2 * (Snew-Sold); Sold = Snew; }
motor[motorB]=v+u; motor[motorC]=v-u; wait1Msec(1); } } task main() { float u, k1 = 2, k2 = 10, a = 0.2, Snew;
int v = 50, d = 8, Sold, L; Snew=Sold=L=SensorValue[S1];
while(true) { Snew=(1-a)*Snew+a*SensorValue[S1]; if (Snew>L*2) { u=v*d/L; Sold = L * 2; } else {u = k1 * (Snew-L) + k2 * (Snew-Sold);
Sold = Snew; }
motor[motorB]=v+u; motor[motorC]=v-u; wait1Msec(1); } }
```

Фільтрування даних стає особливо актуальним, якщо на їх основі потрібно приймати рішення про подальші дії в довгостроковій перспективі [37]. Наприклад, побачивши отвір, зупинитися чи повернути назад. Достатньо однієї перешкоди, щоб робот зупинився не там. Тому фільтри, хоч і загальмовують реакцію робота, але роблять її більш стабільною та передбачуваною.

1.5 Висновки до розділу

У цьому розділі було розглянуто та проаналізовано стан сучасных роботів та їх розвиток. Огляд показав, що роботи стають невд’ємною частиною нашого життя та використовуються в багатьох сферах. Основне їх призначення – зробити наше життя комфортнішим, покращити умови праці, звільнити «руки» від складних робочих процесів та збільшити продуктивність. Роботи найчастіше зустрічаються у промисловості, де з їхньою допомогою вдалося повністю автоматизувати більшість виробничих завдань. Крім того, вони все більше задіяні у військовій галузі, медицині, сфері обслуговування та споживчому секторі.

Також було розглянуто способи програмування робота для пересування. Проаналізовано навігаційні системи. Виявлено, що отримання бажаного результату залежить від вбудованих систем робота, які повинні проаналізувати навколишнє середовище, визначити перепони і побудувати найбільш вдалий маршрут. Задати швидкість і напрямок відстежуючи постійно одержувані дані з датчиків для коректного позиціонування на місцевості і виконання поставлених завдань. Було виявлено, що штучний інтелект стає невід’ємною частиною роботизації.

2 ОБГРУНТУВАННЯ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ДОСЛІДЖЕННЯ МОДЕЛЕЙ ПОВЕДІНКИ МОБІЛЬНИХ РОБОТІВ ПРИ РОЗПІЗНАВАННІ ПЕРЕШКОД

З робототехнікою пов'язано щонайменше сотні інших науково-технічних напрямів, насамперед штучний інтелект (ШІ), з урахуванням якого виробляється система управління роботами. В ШІ також входить півтора десятки наукових напрямів, у тому числі машинний зір [38].

Штучний інтелект (ШІ) - це система або машина, яка здатна імітувати людську поведінку для виконання певних завдань і може поступово навчатися, використовуючи отриману інформацію [39].

ШІ має безліч втілень, наприклад:

1. Чат-боти використовують ШІ, щоб швидше аналізувати звернення замовників та давати відповідні відповіді;
2. «Розумні помічники» використовують ШІ, щоб витягувати інформацію з великих наборів даних у довільній формі та оптимізувати планування;
3. Механізми рекомендацій автоматично підбирають користувачам схожі телепрограми на основі переглянутих раніше.

Сьогодні термін «ШІ» широко використовується для позначення додатків для складних завдань, які раніше могли виконувати лише люди, наприклад, обслуговування замовників або гра в шахи. Нерідко його використовують як синонім машинного навчання та глибокого вивчення, які насправді є підрозділами науки про штучний інтелект і мають свою специфіку [40]. Наприклад, машинне навчання фокусується на створенні систем, які навчаються та розвиваються шляхом обробки та аналізу даних. Важливо розуміти, що якщо машинне навчання завжди має на увазі використання ШІ, то ШІ далеко не завжди має на увазі машинне навчання.

2.1 Історія створення та розвитку ШІ

Ідеї створення машин, що мають свідомість, виникали ще в Стародавній Греції. Вчені створювали механізми, що замінюють людську працю, наприклад, у 17 столітті Паскаль винайшов першу механічну цифрову обчислювальну машину, у 19 столітті Джозеф-Марі Жаккард створив програмований ткацький верстат з інструкціями на перфокартах [41]. У 1937 році Алан Тьрюнінг оприлюднив свій винахід - універсальну машину Тьюринга, в 1939 році в Нью-Йорку були представлені перша механічна людина Electro із собакою Sparco [42].

Однак можливість розробляти програми, що виконують складні інтелектуальні завдання, з'явилася лише після появи сучасних комп'ютерів після Другої світової війни. У 1950-х роках вчені з різних областей почали замислюватись про можливість створення штучного мозку. Тоді дослідження в галузі неврології показали, що мозок є нейронною мережею, а А. Тьюнінг припустив, що будь-який вид обчислень можна уявити в цифровому вигляді, і в 1951 році була створена перша нейронна мережа SNARC аспірантом Марвіном Мінськ. До 1950 року А. Тьюрінг розробив тест, визначальний рівень схожості дій машини зі свідомістю людини, згодом названий тестом Тьюринга. Назва «штучний інтелект» уперше була використана на Дартмутській конференції у 1956 році, тоді ж і з'явилася наукова дисципліна «Дослідження штучного інтелекту» [43].

Згодом було створено безліч машин, які розуміють мову людини, які вміють підтримувати бесіди на задані теми, роботів, які грають у настільні ігри: знаменитий матч між комп'ютером та Каспаровим у шахах закінчився перемогою машини. Наразі штучний інтелект займає важливу позицію у розвитку науки, особливо в рамках концепції Інтернету речей, адже недостатньо лише збирати дані, необхідно їх обробляти, аналізувати та діяти в тих випадках, коли людина цього зробити не може [44].

2.2 Важливість штучного інтелекту

ШІ дозволяє автоматизувати повторювані процеси навчання та пошуку за рахунок використання даних. Однак ШІ відрізняється від роботизації, в основі якої лежить застосування апаратних засобів. Мета ШІ — не автоматизація ручної праці, а надійне та безперервне виконання численних великомасштабних комп'ютеризованих завдань [45]. Така автоматизація вимагає участі людини для початкового налаштування системи та правильної постановки питань.

ШІ адаптується завдяки алгоритмам прогресивного навчання, щоб подальше програмування здійснювалось на основі даних. ШІ виявляє в даних структури та закономірності, які дозволяють алгоритму освоїти певну навичку: алгоритм стає класифікатором або предикатором. Таким чином, за тим же принципом, за яким алгоритм освоює гру в шахи, він може навчитися пропонувати потрібні продукти онлайн. При цьому моделі адаптуються принаймні надходження нових даних. Зворотне поширення - це метод, який забезпечує коригування моделі за допомогою навчання на базі нових даних, якщо початкова відповідь виявляється неправильною.

ШІ здійснює більш глибокий аналіз великих обсягів даних за допомогою нейромереж з безліччю прихованих рівнів. Декілька років тому створення системи виявлення шахрайства з п'ятьма прихованими рівнями було практично неможливим. Все змінилося з колосальним зростанням обчислювальних потужностей та появою «великих даних». Для моделей глибокого навчання потрібна величезна кількість даних, оскільки саме на їх основі вони й навчаються [46]. Тому що більше даних, то точніше моделі.

ШІ дозволяє отримати максимальну користь із даних. З появою алгоритмів, що самонавчаються, самі дані стають об'єктом інтелектуальної власності. Дані містять у собі потрібні відповіді - потрібно лише знайти їх за допомогою технологій ШІ. Оскільки зараз дані відіграють значно важливішу роль, ніж будь-коли раніше, вони можуть забезпечити конкурентну перевагу.

2.3 Технології штучного інтелекту

2.3.1 Комп'ютерний зір

Це обробка візуальної інформації для здобуття знань. Базове завдання всередині цієї технології – детектування об'єкта на зображеннях та відео, тобто усвідомлення того, що на одній фотографії у кутку зображено автомобіль, а на іншій – комп'ютер, клавіатура та телефон. У робототехніці результати виявлення об'єктів дають роботу розуміння, що як робити, а також сприяють його навчанню [47].

Логічним продовженням детектування є трекінг, тобто спочатку об'єкт виявлено, потім починається відстеження його переміщень. Роботам це потрібно, щоб розуміти візуальну сцену та вчитися прогнозувати дії інших об'єктів, що є незамінним, наприклад, для безпілотних автомобілів [47-2].

Інші завдання комп'ютерного зору - це сегментація зображення (розуміння, де підлога, де стіна, а де двері) та оцінка глибини. Останнє має на увазі розуміння відстані до того чи іншого об'єкта і вирішується відновленням тривимірної геометрії серії двовимірних фотознімків [47-3].

Приклади зору робота [47-4]:

1. Детектування об'єктів. Дозволяє знаходити у сфері бачення об'єкти заданих типів.
2. Трекінг об'єктів. Дозволяє відстежувати переміщення об'єкта або об'єктів у сфері бачення.
3. Сегментація. Дозволяє попіксельно визначати вміст галузі бачення.
4. Оцінка глибини. Дозволяє визначати перешкоди на шляху та відстані до них із використанням комп'ютерного зору.

2.3.2 Обробка природної мови

Комунікація з людиною неможлива без розуміння її мови. Фахівці в області ШІ розбирають частинами окремі морфеми, навіть емоційне забарвлення слів у тексті, зашиваючи це в програму. Роботи потребують

таких технологій, для них це як діалогове вікно з людиною, причому йдеться не просто про розуміння, а й про реакцію у відповідь і навчання новим поняттям [48].

2.3.3 Мовна аналітика

Якщо обробка мови стосується текстової інформації, то мовна аналітика – звукова. Насамперед це розпізнавання мови, яке до 2019 року вже міцно увійшло у побут людей. Наступний крок - синтез мови, вдосконалення голосових якостей самого робота та/або програми до рівнів людського спілкування [48-2].

2.3.4 Прийняття рішень

Інакше цю технологію можна назвати автоматизацією процесів, коли вони відбуваються без участі людини. Оскільки знову ж таки йде мова про слабкий ШІ, заточене під вирішення окремих завдань, технології прийняття рішень є чи не найзрозумілішими за своїм призначенням. Автори огляду виділяють кілька сфер застосування таких технологій [48-3]:

1. Навігація, наприклад обхід перешкод, запам'ятовування та облік пройденого шляху, локалізація себе у просторі;
2. Навчання шляхом демонстрацій, коли робот заучує показані візуально чи механічно дії;
3. Емоційна взаємодія, для якої машині потрібно розуміти настрій людини, яка стоїть перед тобою, накладати її на свої особливості «характера» і видавати результат у вигляді «міміки» або «емоцій»;
4. Автоматизація машинного навчання, тобто зниження участі у ньому людини, частковий переведення на самонавчання.

Зрозуміло, такі технології повинні застосовуватися разом з іншими: самостійна навігація разом із комп'ютерним зором, а емоції – разом із мовною аналітикою.

2.3.5 Рекомендаційні системи

Віддалено ця технологія схожа на прийняття рішень, але аналітики [48-4] виділили її окремим пунктом. Причина – потенціал широкого застосування рекомендаційних систем у сервісній робототехніці. Йдеться про пропозицію товарів та послуг, таргетовану рекламу, добірку кінофільмів та музыки. Щодо роботи технологія може призвести до поширення роботів-офіціантів або продавців-консультантів.

2.4 Сьогодення та майбутнє штучного інтелекту

Багато із зазначених вище технологій вже застосовують у робототехніці, причому не тільки у прототипах, а й у масовому виробництві. Найбільший шлях на даний момент пройдено в галузях комп'ютерного зору та обробки природної мови [49] – іншими словами, у розпізнаванні візуальної та текстової інформації.

Вже зараз існують роботизовані системи, які успішно застосовують ті чи інші напрацювання в галузі штучного інтелекту [49-2].

1. Самокеровані автомобілі. Поки що це саме самокеровані, а не безпілотні транспортні засоби. За законом водій все одно необхідний, хоча значну роботу зі сприйняття та оцінки навколишньої дійсності проводить саме машина;
2. Промислові роботи. На виробництві вони застосовуються вже досить довго (наприклад, високоточні верстати або маніпулятори для збирання машин), але технології ШІ почали проникати сюди нещодавно, наприклад машинне навчання роботів, покликаних коригувати роботу сервомоторів, або використання

комп'ютерного зору для оцінки того, як краще упаковати продукт;

3. Кухонні роботи. Комп'ютерний зір допомагає їм визначити місцезнаходження інгредієнтів, посуду та скласти план приготування страви.

У майбутньому розвиток робототехніки відбуватиметься насамперед за рахунок ширшого та глибшого впровадження ШІ, а не вдосконалення матеріально-технічної бази, упевнені автори огляду. Перспективи розвитку ринку вони поділяють на короткострокові та довгострокові, щоправда, конкретних дат не називають.

Короткострокові інновації:

1. Захоплення об'єктів та маніпуляція ними буде доведено до рівня дій людини [49-3];
2. Мобільність роботів, подолання ними перешкод також зрівняються за можливостями з людськими вміннями [49-3];
3. Розмова з роботом буде невідмінна від розмови з людиною [49-3];
4. Витрати та час на програмування роботів скорочуватимуться, що зробить їх самих дешевшими, а впровадження автоматизації — ширшим [49-3].

Довгострокові інновації:

1. За умовчанням кожен робот зможе вирішувати будь-які завдання, притаманні слабкому (вузькоспеціальному) ШІ [49-4];
2. У рамках вирішення своїх завдань роботи стануть повністю автономними, тоді як вихід за їх межі вимагатиме втручання людини [49-4];
3. Безперервний обмін інформацією та якимись вдалими рішеннями між роботами прискорить процес самонавчання [49-4];
4. Роботи почнуть не просто спілкуватися, як люди, вони зможуть планувати поведінку з урахуванням можливого ефекту на оточуючих, які по суті вироблять соціальний інтелект [49-4];

5. Завдяки технологіям ШІ роботи отримають не просто базові знання з певного виду діяльності, але й вважатимуться висококласними фахівцями, наприклад як продавці або медсестри [49-4].

2.5 Аналіз можливості використання ШІ при створенні ігрових додатків

Замість того щоб дізнатися, як найкраще перемогти гравця, ШІ в відеоіграх призначений зовсім для іншого. Він необхідний для поліпшення ігрового досвіду геймерів [50].

Найпоширеніша роль ШІ в відеоіграх - управління неігровими персонажами, і розробники часто використовують різні трюки, щоб NPC виглядали більш розумними. Один із широко використовуваних алгоритмів називається кінцевим автоматом (FSM або finite-state machine) [50-2]. Його ввели в розробку відеоігор в 1990-х роках. У FSM-алгоритмі розробник узагальнює всі можливі ситуації, з якими може зіткнутися ШІ, а потім програмує конкретну реакцію для кожної з них [50-3]. Наприклад, в шутерах штучний інтелект атакує, коли з'являється гравець, а потім відступає, коли його власний рівень здоров'я стає занадто низьким як видно з рис. 2.1.

Рисунок 2.1 – Спрощена схема алгоритма FSM

У прикладі алгоритму FSM NPC може виконувати чотири основні дії у відповідь на можливі ситуації: пошук допомоги, ухилення, блукання та напад. Багато відомих ігор, наприклад Battlefield, Call of Duty і Tomb Raider, включають успішні приклади штучного інтелекту на основі FSM-алгоритму.

Більш просунутий метод, який використовують розробники підвищення персоналізованого ігрового досвіду, — алгоритм дерева пошуку Монте-Карло (MCTS чи Monte Carlo Tree Search). Алгоритм MCTS був створений для запобігання аспекту повторюваності, який є у FSM-алгоритмі [50-4]. MCTS-алгоритм спочатку обробляє всі можливі ходи, доступні NPC у

конкретний час. Потім для кожного з цих можливих ходів він аналізує всі дії, якими міг би відповісти гравець. А далі знову повертається до оцінки NPC вже на основі інформації про вчинки гравця.

Цей алгоритм штучного інтелекту використовувала компанія IBM при створенні Deep Blue - першого шахового суперкомп'ютера, який 11 травня 1997 року увійшов в історію і виграв матч із шести партій у чемпіона світу з шахів Гарі Каспарова [50-5].

Подібний алгоритм застосовують і у багатьох стратегічних іграх [50-6]. Але оскільки можливих ходів набагато більше, ніж у шахах, розглянути їх всі просто не вдасться. У таких іграх алгоритм MCTS буде випадково вибирати деякі з можливих ходів. Завдяки цьому дії NPC стають набагато більш непередбачуваними для гравців.

У такій комп'ютерній грі, як Civilization, існує величезна кількість варіантів подій, доступних комп'ютерному противнику. Будівництво дерева для кожного можливого вибору та сценарію зайняло б дуже багато часу. Саме тому, щоб уникнути таких величезних обчислень, алгоритм MCTS випадково вибирає кілька можливих варіантів [50-7]. У результаті гра забирає менше ресурсів системи, причому ШІ в ній все ще здатний дивувати гравців як видно з рис. 2.2.

Рисунок 2.2 – Спрощена блок-схема використання алгоритма MCST в іграх

2.6 Ігровий штучний інтелект сьогодні

Вимоги розробників ігор стали значною мірою задовольнятися штучним інтелектом, який ми сьогодні не вважаємо таким розумним. Відсутність великих, помітних стрибків у розвитку ігрових ШІ пояснюється тим, що алгоритми, що лежать в його основі, не зазнали радикальних змін.

Сучасні ігри все ще оперують старими фундаментальними концепціями і методами в плані ШІ, але використовують їх у великих масштабах і з перевагами обчислювальної потужності комп'ютерів.

Гумби з гри Super Mario, боси в Dark Souls 3, вороги з Rogue і Dead Cells - всі вони використовують одні і ті ж методи.

Боси з гри Dark Souls 3 рухаються з неймовірною швидкістю, вони запрограмовані передбачити безліч поширених помилок, яких допускають гравці, але використовують такий самий алгоритм поведінки ШІ, як і Гумби в Super Mario [50-8], яка вийшла на кілька десятків років раніше.

Більшість дій ворожих ШІ все ще може систематизувати і передбачити навіть непересічний геймер. Донедавна сучасний ігровий штучний інтелект був здатний пробити собі шлях до впевненої перемоги лише у дуже вузьких областях. Наприклад, у шахах.

У лабораторії DeepMind, що належить Google, в дослідному відділі ШІ Facebook та інших підрозділах по всьому світу старанно вчать ШІ грати в більш складні відеоігри. Це включає все - від китайської настільної гри і класичних проектів Atari, до просунутих кіберспортивних дисциплін на кшталт Dota 2 і CS:GO [50-9].

Також для покращення ігрового ШІ намагаються застосовувати нейронні мережі, нехай і як експеримент. Є кілька дуже відомих недавніх прикладів, одним із яких став ШІ, який переміг професійну команду з Dota 2.

У 2019 році в Сан-Франциско пройшов чемпіонат OpenAI Five, під час якого відбулася зустріч ШІ з п'ятьма кіберспортсменами з команди OG - і люди програли [50-10].

Боти OpenAI навчалися з підкріпленням та незалежно один від одного. Тобто вони потрапили у гру без попереднього програмування, були змушені навчатися методом спроб та помилок. Співзасновник та голова OpenAI Грег Брокман заявив, що за десять місяців свого існування штучний інтелект провів 45 тисяч років ігрового процесу у Dota 2 [50-11].

Після успіху OpenAI деякі люди підняли питання про те, чи може ШІ перемогти гравців в стратегіях реального часу (RTS), таких як StarCraft і

Warcraft. Коротка відповідь: так, може. Але з точки зору можливих ходів і елементів управління RTS набагато складніше.

У ШІ є кілька переваг над людьми - наприклад, здатність до багатозадачності і реагування на що-небудь з блискавичною швидкістю. Тому в деяких іграх розробникам ШІ навіть доводилося свідомо занижувати його можливості, щоб поліпшити ігровий досвід геймерів [50-12].

Сучасні розробники прагнуть не до створення максимально складного ШІ, а скоріше до його вдалим застосуванням всередині систем гри, щоб домогтися так званого Емерджентні геймплея.

В грі Red Dead Redemption 2 від студії Rockstar, яка дозволяє гравцям взаємодіяти з неігровими персонажами тисячами найскладніших способів, що викликають різні реакції в залежності від цілого ряду деталей. Наприклад, від капелюхи, яку ви носите, або від наявності на вашому одязі плям крові. Світ тут настільки складний і пророблений, що різні гравці можуть пережити різні події.

Так само гра Dwarf Fortress використовує величезну кількість геймплейних систем - від процедурно згенерованих рівнів ерозії до різних станів настрою і схильностей жителів-гномів, - щоб створювати унікальні і химерні ситуації.

Також слід згадати гру Middle-Earth: Shadow of Mordor, випущену в 2014 році студією Monolith Productions. Що робить цю гру дійсно особливою і виділяється на тлі інших, так це система Nemesis.

Замість звичайного ШІ, які іноді безглуздо бродять і чекають гравця, вороги в Середзем'я по-справжньому еволюціонують і перетворюються в дійсно небезпечних супротивників. «Тінь Мордора» в якійсь мірі створює штучну соціальну пам'ять - форму інтелекту, яку ми зустрічаємо в реальному житті, але в іграх - майже ніколи. Середзем'я переконує гравців, що їх вороги думають про них на іншому, особистому, рівні - вони пам'ятають, вони ненавидять, а також називають їх по іменах.

Спочатку система розроблялася як щось простіше. Ворожі воїни мали пам'ятати зустрічі з гравцями та реагувати відповідним чином серією

глузувань. Але з розвитком ідеї амбіції розширювалися. Команда ввела елемент просування: ворожі солдати отримують більше влади і просуваються кар'єрними сходами, перемагаючи гравця та інших капітанів.

Nemesis передбачає зради, страти та сутички - гравець може виявити їх і навіть вплинути на результат ситуації. Завдяки цьому кожна битва у грі може бути унікальною. Але найголовніше — внутрішньоігрове суспільство, яке функціонує без головного героя та живе своїм життям. Кожен персонаж створений із ряду випадково призначених сильних та слабких сторін, а також змінних якостей, таких як моральний дух та дисципліна. Зі скромного набору атрибутів команда змогла створювати практично нескінченний ряд ворогів [50-13].

Студія Monolith Productions розробила гру F.E.A.R.: First Encounter Assault Recon, яку випустили у 2005 році. Незважаючи на багато подальших інновацій, штучний інтелект F.E.A.R. все ще вважається у деяких колах стандартом для шутерів від першої особи.

Одним із ключових питань, які мали розглянути розробники, є поведінка NPC, а саме побудова якоїсь довгострокової стратегії [50-14]. Найчастіше навіть сьогодні вороги у шутерах мають реактивніший характер. Це означає, що вони зосереджуються на дії негайно, не враховуючи те, що відбувалося раніше чи може статися у майбутньому. І виникає проблема, що їхня поведінка тоді не командна, не емерджентна. Вони лише прив'язані до конкретних ситуацій.

У F.E.A.R. завдяки плануванню ШІ існує можливість відокремити дії від цілей. Розробники використовували попередні умови для створення NPC. З такою моделлю можна створити цілу низку унікальних дій для різних типів персонажів у грі. А це, у свою чергу, робить поведінку ворогів більш несподіваною та продуманою.

Alien: Isolation від студії Creative Assembly. Під час створення гри її автори використовували дещо незвичайні методи реалізації ШІ. У грі існує система завдань, яка дозволяє Чужому перебувати у двох основних станах – активному та пасивному. Активний стан – це коли система наказує Чужому

обшукати всю локацію чи певні місця після спрацьовування якогось тригера. А пасивний режим включається, коли рівень загрози знаходиться на піку надто довго, а потім різко зникає. Тоді Чужий самостійно намагається знайти гравця.

Поведінка інопланетянина залежить від заздалегідь прописаного дерева. Монстр має більш ніж 100 вузлів, прихованих у його системі. Але при запуску гри використовується всього 30. Система поступово розблоковує комплексні моделі поведінки - у міру виконання певних умов протягом гри. Це гарантує, що чим більше геймер проводить часу в Alien: Isolation, тим сильніше Чужий починає виявляти нові риси поведінки, щоб постійно дивувати і шокувати. Саме завдяки такій концепції складається враження, що монстр починає вчитися на власному досвіді та, що важливіше, на вчинках гравця [50-15].

2.7 Висновки до розділу

У цьому розділі розглянуто, що штучний інтелект, як и роботизація створені для того, щоб поліпшити життя людини. Його мета надійне та безперервне виконання численних великомасштабних комп'ютеризованих завдань. Детально розглянуті способи навчання штучного інтелекту та взаємодія з дюдиною.

ШІ адаптується завдяки алгоритмам прогресивного навчання, щоб подальше програмування здійснювалось на основі даних. В майбутньому розвиток ШІ в іграх, не буде зосереджено на створенні більш потужних NPC (ботах), щоб вони шукали витончені способи перемоги над гравцями. Замість цього розробка сконцентрується на тому, як створити унікальний ігровий досвід для кожного геймера, саме ШІ який використовується в ігрових двигунах може підняти ігровий досвід на новий рівень.

3 АНАЛІЗ ВИКОРИСТАННЯ ІГРОВИХ ДВИГУНІВ ДЛЯ ДОСЛІДЖЕННЯ МОДЕЛЕЙ ПОВЕДІНКИ МОБІЛЬНИХ РОБОТІВ ПРИ РОЗПІЗНАВАННІ ПЕРЕШКОД

3.1 Історія створення та використання ігрових двигунів

Вперше визначення «двигун гри» з'явилося в середині 90-х років, коли почали з'являтися ігри, схожі на головний шутер на той час – Doom. В той же час у вільному доступі почали з'являтися ігрові двигуни, на основі яких сторонні розробники, і звичайні користувачі могли пробувати писати власні ігри [51].

З тих пір ігрові двигуни ставали все більш складними технічно, довгими та насиченими за своїм програмним кодом. Але при цьому, як і на початку свого існування, вони містять жорстко фіксовані дані:

1. Ігрову логіку;
2. Фізику об'єктів;
3. Правила малювання об'єктів;
4. Геймплей загалом.

«Поверх» двигуна прописуються всі інші елементи гри, і їх багато. Тому навіть при використанні одного і того ж двигуна в результаті віртуальні світи виходять абсолютно різними.

3.2 Аналіз основних понять

Двигун гри (game engine) - це її основне ядро, базове програмне забезпечення, на основі якого будуються всі інші складові гри. Програмний код, який може використовуватися для створення варіацій гри, аддонів до неї або навіть нового ігрового світу [51-2].

Ігрові двигуни спеціалізовані в рамках жанру комп'ютерних ігор. Так, двигун, спроектований для двовимірного файтингу на боксерському рингу,

істотно відрізнятиметься від движка для масової, на багато користувачів гри, шутера від першої особи або стратегії в реальному часі. Але в той же час движки мають суттєві загальні частини - всі тривимірні ігри, незважаючи на жанр, вимагають взаємодії гравця за допомогою клавіатури, геймпада та/або миші, деяку форму тривимірного рендерингу, засоби індикації, як на лобовому склі (наприклад, друк тексту поверх зображення), звукову систему та багато іншого [51-3]. Так, двигун Unreal Engine, незважаючи на те, що був спроектований для шутера від першої особи, успішно використовувався для створення ігор у багатьох інших жанрах, таких як шутер від третьої особи Gears of War, пригодницька рольова гра Grimm або футуристична гонка Speed Star.

3.3 Аналіз вимог до ігрового двигуна при створенні та дослідженні моделей поведінки мобільних роботів при розпізнаванні перешкод

Розглянемо основні жанри ігор та вимоги від движка для кожного з них. Історично шутери від першої особи відносяться до ігор, які найбільш технологічно складні, тому що їм необхідно надати гравцю ілюзію тривимірного світу і робити це для активних дій у реальному часі. Двигуни шутерів від першої особи більше звертають увагу на такі технології, як ефективний рендеринг тривимірних світів, чуйна ігрова механіка контролю та прицілювання, висока точність анімації зброї та рук керованого гравцем персонажа, широкий спектр ручного озброєння, «прощаюча» модель руху гравця та його зіткнення з перешкодами, висока якість анімації та штучного інтелекту неігрових персонажів. При цьому характерні мала масштабованість у розрахованих на багато користувачів іграх (типова підтримка до 64 гравців) і повсюдна орієнтація на ігровий процес deathmatch. Графічні двигуни ігор даного жанру використовують низку оптимізацій залежно від поточного оточення гравця, але водночас пред'являються вимоги з анімації персонажа, аудіо та музики, динаміки твердого тіла, кінематики та інших технологій [51-4].

Двигуни платформерів звертають більше уваги на анімацію персонажа та його аватара, і при цьому їм не потрібно тієї реалістичності, яка притаманна тривимірним шутерам. Для платформерів характерно застосування ряду технологій: безліч способів переміщення (рухомі платформи, сходи, мотузки, підпірки та інші), елементи з головоломок, використання камери, що стежить за персонажем від третьої особи, рендеринг декількох шарів геометрії в поєднанні з системою зіткнень об'єктів, та інші.

Файтинги орієнтовані на багату анімацію, точність ударів, можливість завдання складних комбінацій за допомогою кнопок та/або джойстика тощо. Анімаційні персонажі пред'являють вимоги двигунам з високої деталізації, додатково двигуни забезпечують можливість зміни та додавання спецефектів (шрамів після ударів, виступ поту тощо), а також надаються можливості симуляції зачіски, одягу та інших елементів.

Автосимулятори можуть бути різними і тут є низка підджанрів. Графіка таких ігор орієнтована на «коридорність» та кільцеві треки, і тому двигуни більше звертають увагу на деталізацію машин, треку та безпосереднє оточення. Як наслідок, використовуються технології для рендерингу далеких фонових об'єктів (відображаються двовимірно), трек часто поділяється на кілька секторів, всередині яких проводиться оптимізація рендерингу. У разі руху тунелями або іншими «тісними» місцями використовуються техніки для того, щоб камера з виглядом від третьої особи не перетиналася з фоновою геометрією. Використовувані структури даних і штучний інтелект орієнтуються вирішення завдань машин неігрових персонажів, як-от пошук шляху та інших технічних проблем.

У стратегій реального часу немає високих вимог до графіки і тому двигун орієнтується те що, що відображає юнітів у низький роздільній здатності, але у своїй він може бути здатний працювати з великою кількістю юнітів одночасно. Окремі особливості є в інтерфейсі взаємодії гравця та елементів управління, до яких входять інструменти роботи з групами юнітів (виділення за площею, управління) та ряд меню та панелей інструментів, що

містять команди управління, елементи спорядження, вибір типів юнітів та будівель тощо.

Масові розраховані на багато користувачів ігри вимагають наявності великого ігрового світу і можливості одночасної присутності і взаємодії великого числа гравців. Локальні завдання, що вирішуються двигуном, схожі на ті, що є в іграх інших жанрів, але особливістю жанру є орієнтація та опрацювання програмного забезпечення серверів, які повинні зберігати стан світу, управляти підключенням та відключенням гравців, надавати внутрішньоігрові чати, способи взаємодії голосом тощо.

3.4 Огляд та класифікація ігрових двигунів

В роботі було проведено аналіз існуючих ігрових двигунів. Розглянемо їх переваги та недоліки з огляду на створення програмно-апаратних засобів дослідження моделей поведінки мобільних роботів при розпізнаванні перешкод.

3.4.1 Аналіз ІД 4A Engine

Один з найпопулярніших двигунів для шутерів і екшенів, написаний українськими розробниками - вихідцями з GSC Game World. Ця платформа використовується тільки для внутрішніх потреб компанії і не доступна для інших розробників ні на якій основі (ні платно, ні безкоштовно). З технічної точки зору вона є покращеним X-Ray з доопрацьованим PhysX, тесселяцією для поліпшення графіки, а також повною руйнівністю об'єктів.

Двигун забезпечує тривимірне позиціонування звуку, динамічне освітлення, безліч умов бою, відмінні умови для скриптування. Крім того, в ньому є система аналізу топології ШІ, можливість наділити персонажів зором, слухом та іншими почуттями, а також задати їхню групову поведінку. Загалом, на основі цього двигуна можна писати справді складні шутери та екшени [52].

Попри великої кількості переваг у цього двигуна є великий недолік – він створеный сутьо для внутрішніх потреб компанії і не доступний для інших розробників.

3.4.2 Аналіз ІД Anvil

Двигун, написаний суто для внутрішнього використання - у грі Assassins Creed. В основі використовується код C++, з відмальовуванням в ZBrush і 3ds Max та створенням фізики в не менш відомому Havok. Двигун не найлегший, а тому вимагає систем з хорошими ресурсами в плані продуктивності, зате він забезпечує реально круту картинку з деталізованою анімацією, реалістичними погодними умовами, великою кількістю персонажів в одній сцені (до трьох тисяч!), а також активним і досить таки розумним NPC як просто у грі, так і під час боїв [53].

Anvil ще один двигун написаний для внутрішнього використання, це досі залишується одним з ключевих факторів чому двигун не підходить для поставленої задачі.

3.4.3 Аналіз ІД Creation Engine

Перше і одне з основних переваг цієї платформи - підтримка величезних локацій з детальним промальовуванням всіх об'єктів на ній, а також можливістю вільного і швидкого переміщення, без додаткового підвантаження текстур.

Крім того, розробники Creation Engine приділили увагу ШІ (штучному інтелекту), зробивши його практично досконалим, роботі з водою та снігом, завдяки чому вони виглядають вкрай реалістично, а також анімації та заснованому на фізиці рендерингу [54].

3.4.4 Аналіз ІД CryEngine 4

Німецька студія Crytek продовжує оновлювати та осучаснювати своє ігрове ядро, при цьому, як і колишні версії, CryEngine 4 поширюється практично безкоштовно – з мінімальною оплатою. При цьому за своїми можливостями він не урізаний, і ви можете чудово бачити це в іграх серії FarCry. CryEngine 4 забезпечує відображення величезних безшовних локацій, інверсної кінематики транспорту та персонажів, відмінну імітацію нетвердих об'єктів, параметри штучного інтелекту, що настроюються, звучання формату 5.1, а також безліч інших переваг. Загалом, це відмінна платформа і для досвідчених розробників, і для початківців [55].

CryEngine є одним з небагатьох двигунів, найбільш підходячих до поставлених цілей, однією з переваг є імітація різних об’єктів.

3.4.5 Аналіз ІД id Tech

Id Tech існує вже в сьомій версії і протягом усього існування, що розповсюджується на повністю безкоштовній основі. При цьому не варто думати, що двигун урізаний або простий - на його основі створені такі хіти, як Wolfenstein, Quake, Rage, Doom. Тут якісно зроблено відображення текстур, є окремий потік для обробки кожної складової двигуна і навіть є півтіні для затінення ділянок у кадрі [56].

Хоча id Tech і поширується на безкоштовній основі, що дає йому переваги перед іншими двигунами, але нажаль він заточений на шутери від першої особи.

3.4.6 Аналіз ІД Frostbite

Двигун використовується у багатьох іграх цієї студії, включаючи Battlefield, FIFA, DragonAge, PayBack. Простіше кажучи, це платформа-універсал, з урахуванням якої написані екшени, РПГ, гонки, спортивні симулятори тощо. Другий дивовижний момент полягає в тому, що у двигуна практично немає недоліків, зате є безліч переваг: безліч пост-ефектів,

відмінна руйнівність об'єктів, тривимірні та двовимірні спецефекти, реалістичні текстури і навіть окремий ігровий редактор для роботи з шейдерами та дрібними деталями [57].

Frostbite є одним з небагатьох двигунів які можна використовувати, як було сказано – він універсал і пидходить для великої кількості задач.

3.4.7 Аналіз ІД IW Engine

Платформа, що використовується в серії шутерів Call of Duty, і це один з небагатьох двигунів, в якому змодельована система невагомості. Крім того, це ядро дозволяє використовувати змінні текстури, безліч спецефектів, динамічні погодні умови, створювати багатошарові тексти, імітувати контузію персонажа, налаштовувати перегрів ствола в залежності від швидкості стрільби або температури навколо персонажа та багато іншого. При цьому двигун добре оптимізований, так що відрізняється високою продуктивністю і не дуже вимогливий до ресурсів [58].

IW Engine має в своєму арсеналі досить цікаві можливості, яких немає в інших двигунах, але цей двигун більше підходить для ігор жанру шутер.

3.4.8 Аналіз ІД Rage Engine

Студія Rockstar Games також написала двигун для свого використання - Rage Engine. Причому багато компонентів цього коду написані з нуля співробітниками компанії, а взагалі движок описує практично всі складові гри: в ньому є звукова, графічна, анімаційна та мережева складові, штучний інтелект, власна скриптова мова та модулі для роботи в онлайні. Як і в багатьох сучасних ігрових ядрах, тут є підтримка величезних локацій без дозавантаження, включаючи захід у розташовані на них будівлі, підтримка динамічної погоди та навіть численних видів транспорту. Якщо говорити про реальні приклади використання цього движка, то варто згадати в першу

чергу недавно вийшов Red Dead Redemption 2, а також культовий Max Payne 3 [59].

Без сумніву Rage Engine іноваційний двигун який включає в себе багато функцій, але він створенний виключно для внутрішнього використання.

3.4.9 Аналіз ІД Source

Важливими складовими цієї платформи є добре опрацьована лицьова анімація, просунутий ШІ, що дозволяє суперникам збиратися в угруповання, а також якісна робота з шейдерними ефектами. Крім того, у цьому коді передбачена робота з динамічними джерелами світла, включаючи автоматичне затемнення, відмінну руйнівність об'єктів та круту кінематографічну фізику [60]. Хоча Source и має багато переваг, для реалізації завдання він не досить підходе .

3.4.10 Аналіз ІД Unreal Engine 4

Є кілька причин великої популярності UE4:

1. UE4 підтримує велику кількість функцій, завдяки чому можна створити практично будь-яку гру [61].
2. В Unreal Engine 4 є вбудована система візуального скриптингу, яка дозволяє без особливих перешкод вибудовувати ігрову логіку навіть новачкам [61-2].
3. Двигуном можна користуватися безкоштовно: у ліцензійній угоді Unreal Engine зазначено, що поки ваша гра не принесе більше $ 1 000 000 - після цього доведеться платити 5% від доходу [61-3].

Також у Unreal Engine 4 є можливість зробити гру, яка запуститься на всіх популярних платформах: PlayStation, Xbox, Switch, ПК, iOS, Android.

Двигун має величезну спільноту користувачів, які створюють навчальні матеріали, діляться один з одним досвідом і допомагають

вирішувати проблеми. Додатковий плюс великої спільноти – безліч ігрових ассетів, доступних для вільного використання у своєму проекті.

Велика перевага Unreal Engine 4 в універсальності та доступності - його можуть використовувати як досвідчені розробники, так і новачки, які вперше беруться до створення гри. Справа в тому, що UE4 за замовчуванням підтримує відразу дві мови програмування: текстовий C++, в якому потрібно писати рядки коду, і візуальна мова Blueprints, в якій ігрова логіка вибудовується за допомогою пов'язаних між собою блоків. Такий підхід допомагає зробити програмування наочнішим і зрозумілішим для тих, у кого немає досвіду [61-4].

Blueprints розробила компанія Epic Games, щоб дати інструменти програмування людям, які далекі від цього. Тому навчитися працювати з Blueprints дуже просто. Звичайно, необхідно отримати мінімальну теоретичну базу, наприклад, розібратися, що таке змінні та як працює функціональне та об'єктно-орієнтоване програмування [61-5].

Але і цю теоретичну основу можна вивчати, використовуючи Blueprints. І навіть без неї можна зробити щось інтерактивне, наприклад, кнопку, яка відчиняє двері. Мова Blueprints дуже проста в освоєнні, і в ньому є багато рішень, які роблять його застосування зручнішим. Наприклад, колірна індикація різних типів змінних.

Велика перевага Blueprints у тому, що за його допомогою можна швидко зібрати базовий геймплей для гри.

Незважаючи на те, що Blueprints простіше для розуміння, такий скриптинг майже не поступається C++ за функціями є лише кілька рідкісних винятків, в яких Blueprints трохи обмежений.

Таких обмежень замало. Вони впираються або у неможливість змінити вихідний код двигуна UE4 (для цього потрібен C++) і базових класів проекту, або у продуктивність складних математичних розрахунків, наприклад, обробки таблиць даних сотень гравців у MMO [61-6].

Майже все можна продати на Blueprints. Всі принципи та підходи програмування, які використовують у текстових мовах (на кшталт C++), також можна застосовувати у Blueprints.

У результаті, Unreal Engine 4 - один з найпопулярніших двигунів через кілька важливих переваг:

1. Універсальність. UE4 можна використовувати і для PlayStation, і для Switch, і для ПК, – буквально для чого завгодно – це дає потрібну гнучкість при розробці гри;
2. Велика бібліотека ассетів;
3. Простота входу через Blueprints. Щоправда, для професійної роботи все одно доведеться вивчити C++;
4. Гарна задокументованість та підтримуваність, активна спільнота з купою opensource-інструментів;
5. Можливість безкоштовного використання для маленьких проектів.

Володіння Unreal Engine 4 відкриває перед спеціалістом великі можливості. Він може зробити проект поодинці, а може приєднатися до команди. При цьому попередній досвід має не таке вже велике значення — розробник-початківець без особливих проблем зможете навчитися Blueprints, щоб створити власний проект. Це стане чудовою базою для подальшого вивчення C++, знання якого дозволить ще більше заглибитись у створення внутрішньої архітектури гри чи віртуального робота з ШІ.

3.5 Висновки до розділу

Таким чином, огляд сучасних ігрових двигунів показав, що їх використання є актуальним на сьогоднішній день. Детально розглянуті технічні переваги і недоліки деяких ігрових двигунів. Як, показав аналіз, більшість ігрових двигунів створені виключно для внутрішніх потреб студії розробника, і це не дає можливості використовувати їх двигун, також

недоліком є те, що зазвичай певний двигун підходить тільки для певного жанру гри.

Проте було виявлено, що кращим двигуном для поставленої задачі є Unreal Engine, що розробляється та підтримується компанією Epic Games. Рівень якості та кількість корисних можливостей, які пропонуються розробникам, роблять цей двигун дійсно універсальним, та на відміну від інших він бескоштовний та може використовуватися будь ким.

4 РОЗРОБКА МОДЕЛЕЙ ДЛЯ ДОСЛІДЖЕННЯ ПОВЕДІНКИ МОБІЛЬНИХ РОБОТІВ ПРИ РОЗПІЗНАВАННІ ПЕРЕШКОД

В цій роботі будуть використовуватись сценарійї і моделі Unreal Engine для дослідження поведінки мобільних роботів при розпізнаванні перешкод.

Для порівняння розглянемо одну з моделей поведінки робота Unreal Engine та поведінку оператора при проходженні одних і тих же перешкод.

В роботі було створено модель робота який розпізнає перешкоди при переході з точка *А* в точку *Б*. Для цього робота було запрограмовано за допомогою встроєного забобу Unreal Engine Blueprints. Розглянемо поетапто розписаний принцип його дій. Blueprints зору робота для дослідження його поведінки, при визначенні наявності перешкоди реалізований за допомогою вбудованного ШІ в Unreal Engine, початок Blueprints зору робота приведено рис. 4.2:

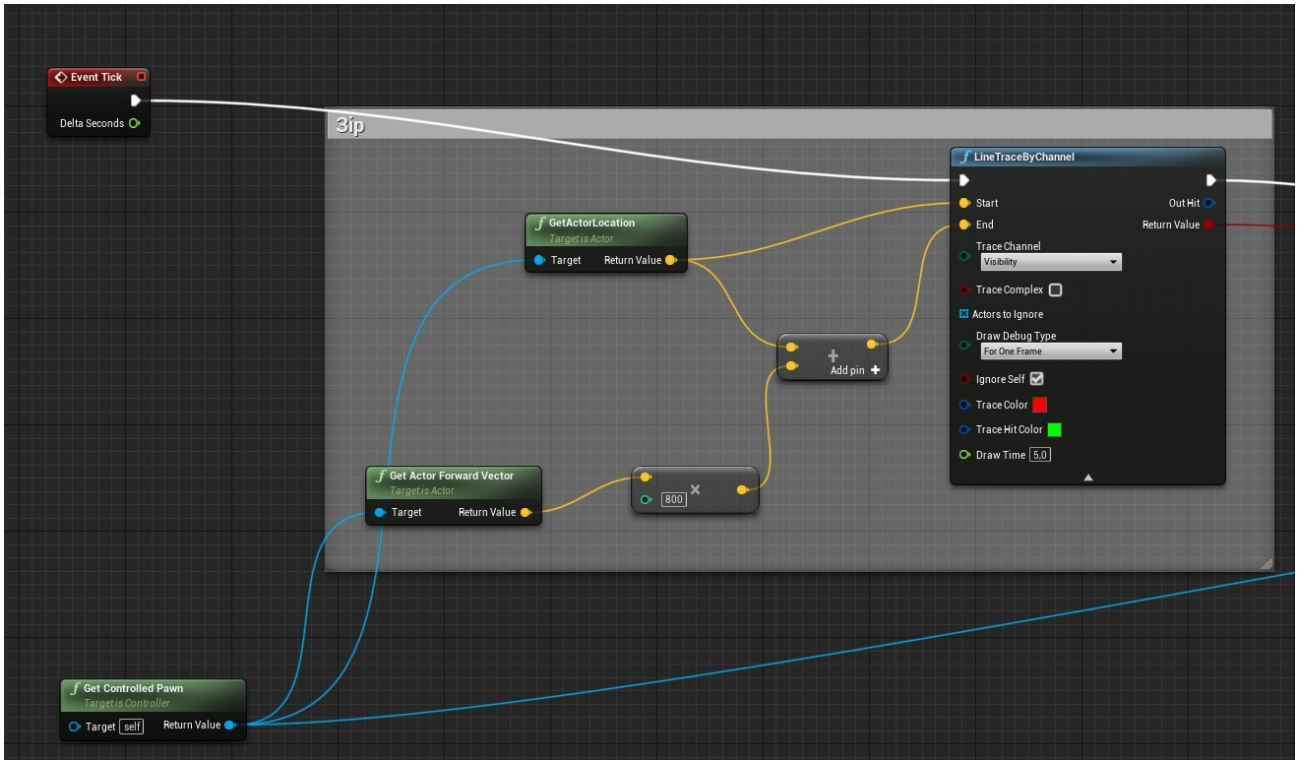


Рисунок 4.2 – Blueprint зору робота для дослідження його поведінки, при визначенні наявності перешкоди

Данний blueprint дає можливість роботу бачити перед собою завдяки ноду LineTcareByChanel, його суть в тому що перед роботом з'являється промінь, в який при попаданні перешкоди роботу йде сигнал про те, що потрібно вживати якихось заходів, для подолання перешкоди.

Наступний крок – це навчання робота, саме для цього було створено blueprint прийняття рішень при появі перешкоди, який зображено на рис. 4.3-рис. 4.4:

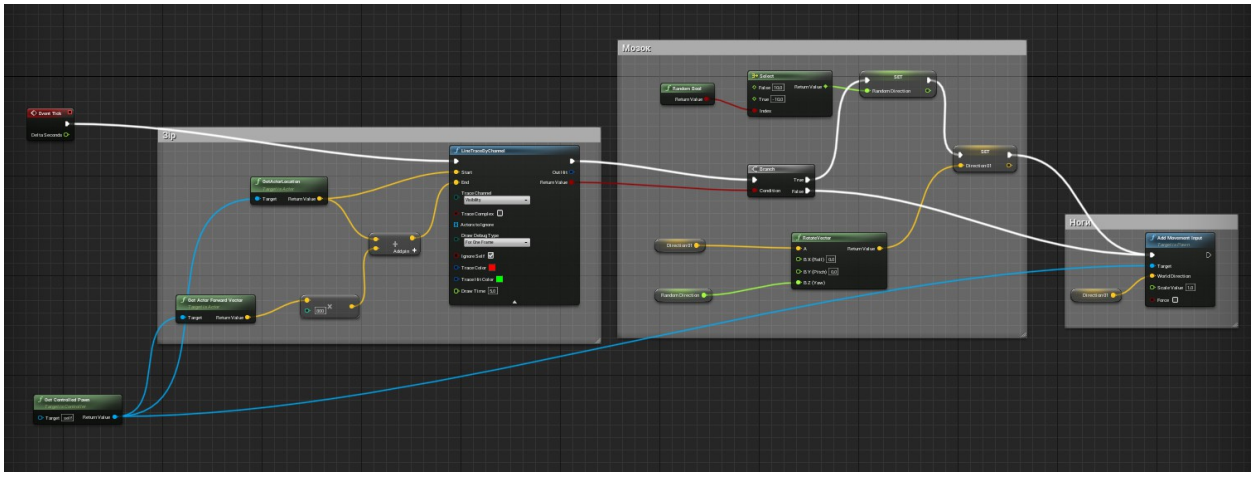


Рисунок 4.3 – Загальний blueprint робота

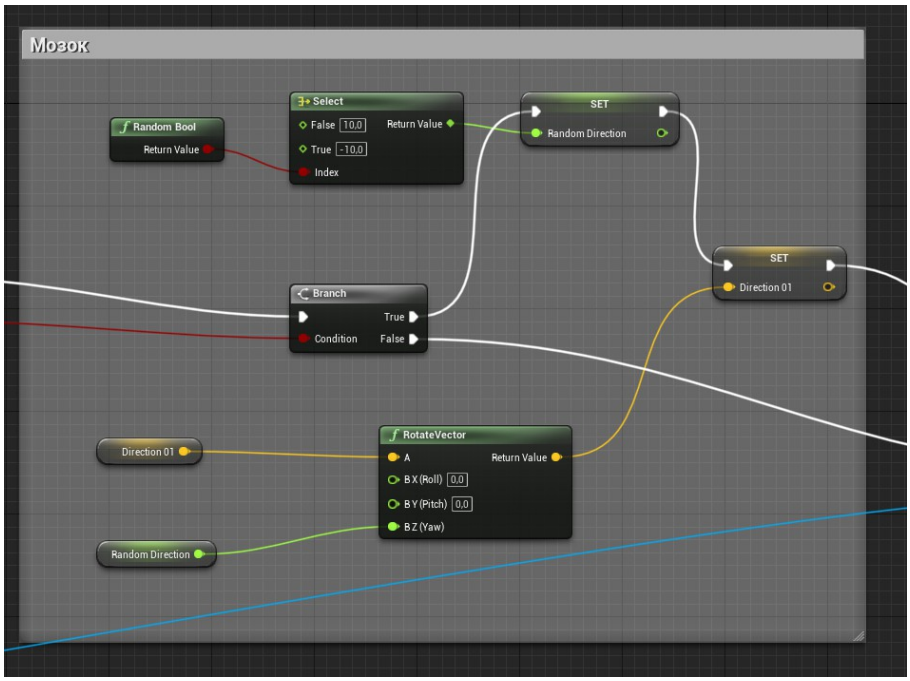


Рисунок 4.4 – Blueprint прийняття рішень при появі перешкоди

На рис. 4.4 представлено blueprint за допомогою якого виконується прийняття рішень роботом, яке слід прияняти рішення при виявленні перешкоди, наприклад, змінити свій напрямок руху.

Наступний этап є створення здатності робота рухатися, цей blueprint показано на рис. 4.5:

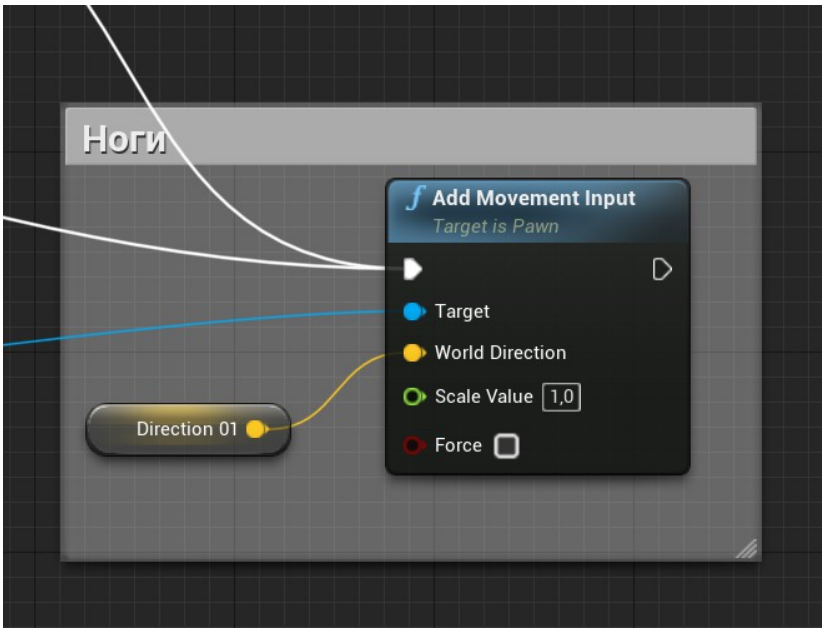


Рисунок 4.5 – Blueprint пересування робота

Взявши два різні сценарія для однакових роботів під керівництвом ШІ та оператора.

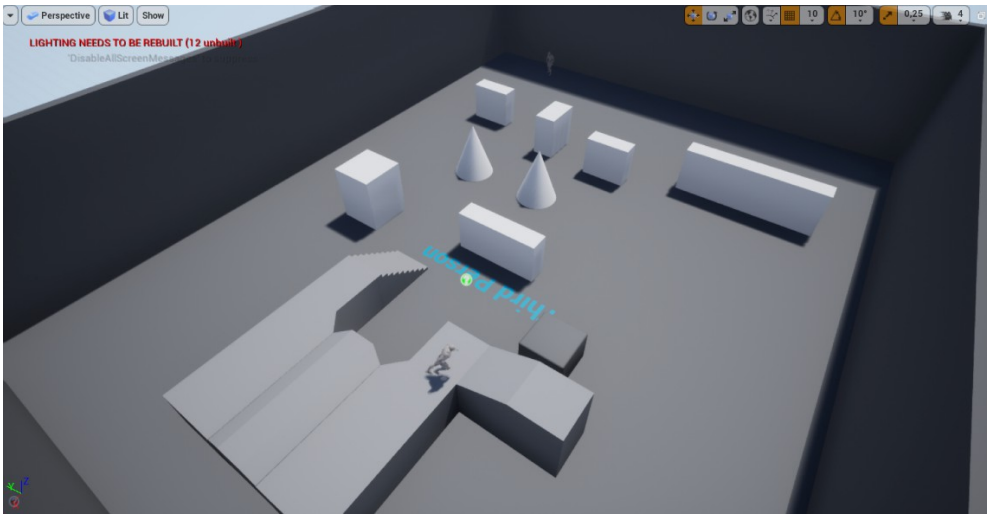
Роботу моделі ШІ Unreal Engine показано на рис. 4.6, ми бачимо пересування робота по полігону, де він знаходить найкоротший шлях до кінцевої точки *Б*:


Perspective
Lit
Show
LIGHTING NEEDS TO BE REBUILT (12 unbuilt)

а


Perspective
Lit
Show
LIGHTING NEEDS TO BE REBUILT (12 unbuilt)

б


Perspective
Lit
Show
LIGHTING NEEDS TO BE REBUILT (12 unbuilt)

в

Рисунок 4.6 – Пересування робота по полігону до точки *Б*:
а – перша перешкода; б – друга перешкода; в – третя перешкода

Як видно з рис. 4.6 робот під управлінням штучного інтелекту проходить полігон, шукаючи для себе найкоротший шлях, щоб якнайшвидше дістатися кінцевої точки *Б*.

Роботу моделі керування роботів 1 - керування оператором (людиною) та 2 - ШІ Unreal Engine показано на рис.4.7. Ми бачимо пересування роботів по полігону, де вони знаходять найкоротший шлях до кінцевої точки *Б*:

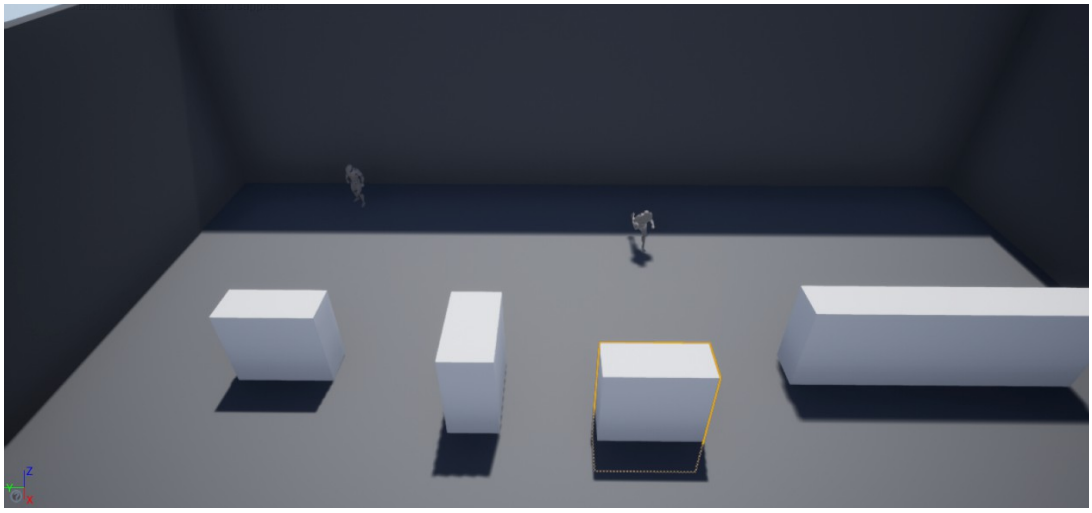


Рисунок 4.7 – Початок шляху роботів

Поки робот під управління оператора дійшов до першої точки, робот під управління штучного інтелекту подолає другу, що демонструє його перевагу перед іншим роботом, як видно з рис. 4.8:

Рисунок 4.8 – Друга контрольна точка

Як видно з рис. 4.9 робот під керівництвом штучного інтелекту все ще зберігає свою перевагу

Рисунок 4.9 – Третя контрольна точка

Як видно із рис. 4.10 робот під управлінням штучного інтелекту завершив завдання раніше робота під управлінням оператора

Рисунок 4.10 – Четверта контрольна точка

Провівши дослідження на їхню швидкість пересування були виявлені наступні результати на чотирьох відібраних точках, які показані на таб. 4.1:

Таблиця 4.1 - Швидкість пересування роботів

Обєкт дослідження	Швидкість проходження в КТ			
	КТ №1	КТ №2	КТ №3	КТ №4
ШІ	2	2,55	3,28	3,52
Оператор	0	1,36	2,41	3,11

Як видно з результатів дослідження ШІ має більшу швидкість виконування задачі ніж робот під керівництвом оператора.

Проведено дослід з двома сценаріями для виявлення кращого алгоритму виконання постевленої задачі, а саме дістатися кінцевої точки уникаючи перешкоди на шляху до неї.

Робот під керуванням ШІ Unreal Engien проходив полігон декілька разів, стабільно притримуючись гарного результата, на відміну від робота під керівництвом оператора (людини), що разу проходячи полігон він вдосконалював свій результат, оскільки набирався досвіду на данному полігоні.

За результатами досліду було виявлено, що робот під управлінням ШІ Unreal Engien виконав задачу швидше, та йому знадобылося меньше часу на навчання на відміну від робота під управлінням оператора.

ПЕРЕЛІК ПОСИЛАНЬ

1. Роботи та нові технології. [Електронний ресурс] URL: https://robroy.ru/ (дата звернення: 10.11.2021)
2. Что таке роботи? [Електронний ресурс] URL: https://rb.ru (дата звернення: 10.11.2021).
3. Що таке ігровий двигун? [Електронний ресурс] URL: https://funduk.ua/ (дата звернення: 13.11.2021).
4. Що таке штучний інтелект? [Електронний ресурс] URL: https://www.oracle.com (дата звернення: 13.11.2021).
5. Важливість штучного інтелекту [Електронний ресурс] URL: https://www.sas.com (дата звернення: 13.11.2021).
6. Алгоритми машинного навчання [Електронний ресурс] URL: https://habr.com (дата звернення: 14.11.2021).
7. Розвиток штучного інтелекту у роботів [Електронний ресурс] URL: https://trends.rbc.ru (дата звернення: 14.11.2021).
8. Історія та розвиток штучного інтелекту [Електронний ресурс] URL: https://iot.ru (дата звернення: 15.11.2021).
9. Штучний інтелект в ігровій індустрії [Електронний ресурс] URL: https://skillbox.ru (дата звернення: 15.11.2021).
10. Сучасні роботи [Електронний ресурс] URL: https://vc.ru (дата звернення: 16.11.2021).
11. Алгоритми пересування роботів [Електронний ресурс] URL: https://intellect.icu/ (дата звернення: 16.11.2021).
12. Перспективи розвитку робототехніки [Електронний ресурс] URL: https://studbooks.net/ (дата звернення: 18.11.2021).
13. A Comparative Example Between The Use Of Pca And Mds For Image Classification / Hernandez, W., Mendez, A., Flor-Unda, O., Camejo, I.M., Kolendovska, M.// IEEE International Symposium on Industrial Electronics, 29th IEEE International Symposium on Industrial Electronics,

ISIE 2020; Delft; Netherlands; 17 June 2020 до 19 June 2020; Volume 2020-June, June 2020, № 9152565, Pages 1353-1358

14. Algorithm For Generating Refined Frequency Estimates In Atmospheric Radio Sounding Systems / Kartashov V., Hernandez W., Hernandez-Balbuena D., M. Kolendovska, Konovalenko O., Melnyk V.// IEEE International Symposium on Industrial Electronics, 29th IEEE International Symposium on Industrial Electronics, ISIE 2020; Delft; Netherlands; 17 June 2020 до 19 June 2020; Volume 2020-June, June 2020, № 9152562, Pages 79-82
15. Application of Fast Frequency Shift Measurement Method for INS in Navigation of Drones / D. Avalos-Gonzalez, D.H. Balbuena, V. Tyrsa, V.M. Kartashov, M. Kolendovska, S. Sheiko, O. Sergiyenko, V. Melnyk, F.N. Murrieta-Rico // IECON 2018 – 44th Annual Conference of the IEEE Industrial Electronics Society. – P. 3159–3164.
16. Avalos-Gonzalez, D., Sergiyenko, O., Hernandez-Balbuena, D., Tyrsa,V., Kartashov V.M., V.,Rivas-Lopes, M., Murrieta-Rico, F.N. Constraints definition and application optimization based on geometric analysis of the frequency measurement method by pulse coincidence// Measurement: Journal of the International Measurement Confederation (USA). 2018, V.126. P. 184-193.
17. Book “Control and Signal Processing Applications for Mobile and Aerial Robotic Systems”, Hardback - Advances in Computational Intelligence and Robotics English. Edited by Oleg Sergiyenko, Moises Rivas-Lopez, Wendy Flores-Fuentes, Julio Cesar Rodríguez-Quiñonez, Lars Lindner. Editorial IGI Global, Hershey, United States, January 2020, 340 páginas. ISBN10 152259924X, ISBN13 9781522599241
18. Cesar Sepulveda-Valdez ; Oleg Sergiyenko ; Vera Tyrsa ; Wendy Flores-Fuentes ; Julio César Rodríguez-Quiñonez ; Fabian Natanael Murrienta-Rico ; Jesús Elías Miranda-Vega ; Paolo Mercorelli ; Marina Kolendovska. "Geometric analysis of a laser scanner functioning based on dynamic triangulation," 2020 IEEE 29th International Symposium on Industrial

Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1398-1403, doi: 10.1109/ISIE45063.2020.9152268. https://ieeexplore.ieee.org/abstract/document/9152268

19. Cuauhtémoc Mariscal-García; Wendy Flores-Fuentes; Daniel Hernández-Balbuena; Julio C. Rodríguez-Quiñonez ; Oleg Sergiyenko. "Classification of Vehicle Images through Deep Neural Networks for Camera View Position Selection," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1376-1380, doi: 10.1109/ISIE45063.2020.9152440. https://ieeexplore.ieee.org/abstract/document/9152440
20. Developing and Applying Optoelectronics in Machine Vision/ O. Sergiyenko, J.C. Rodriguez-Quiñonez, IGI Global, 2016; 341p.
21. Experimental estimation of direction finding to unmanned air vehicles algorithms efficiency by their acoustic emission, /Oleynikov, V., Zubkov, O., Kartashov, V., ...Sheiko, S., Babkin, S.//2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 - Proceedings, 2019, стр. 175-178, 9061337
22. Features of acoustic noise of small unmanned aerial vehicles / Semenets, V.V., Kartashov, V.M., Leonidov, V.I. //Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 2020, 79(11), стр. 985-995
23. Geometric Analysis Of A Laser Scanner Functioning Based On Dynamic Triangulation /Sepulveda-Valdez, C., Sergiyenko, O., Tyrsa, V, Mercorelli, P., Kolendovska, M.// IEEE International Symposium on Industrial Electronics, 29th IEEE International Symposium on Industrial Electronics, ISIE 2020; Delft; Netherlands; 17 June 2020 до 19 June 2020; Volume 2020-June, June 2020, № 9152268, Pages 1398-1403

https://ieeexplore.ieee.org/abstract/document/9152255

https://ieeexplore.ieee.org/document/9161870

24. I. Y. A. Corpus, L.Lindner, O.Sergiyenko. "Transimpedance Amplifier for Laser Scanning System Range Extension," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1421-1426, doi: 10.1109/ISIE45063.2020.9152487. https://ieeexplore.ieee.org/abstract/document/9152487
25. Ivanov, M., Sergiyenko, O., Mercorelli, P., Hernandez, W.c, Rodriguez Quinonez, J.C.d, Katashov V., Kolendovska, M., Iryna, T. Effective informational entropy reduction in multi-robot systems based on real-time TVS. IEEE International Symposium on Industrial Electronics, 2019-June,8781209, c. 1162-1167.
26. Jonathan J. Sanchez-Castro ; Julio C. Rodríguez-Quiñonez ; Luis R. Ramírez-Hernández ; Guillermo Galaviz ; Daniel Hernández-Balbuena ; Gabriel Trujillo-Hernández ; Wendy Flores-Fuentes ; Paolo Mercorelli ; Wilmar Hernández-Perdomo ; Oleg Sergiyenko ; Félix Fernando González-Navarro. "A Lean Convolutional Neural Network for Vehicle Classification," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1365-1369, doi: 10.1109/ISIE45063.2020.9152274. https://ieeexplore.ieee.org/abstract/document/9152274
27. Lindner, L., Sergiyenko, O., Rivas-López, M., (...), Gurko, A., Kartashov, V.M. Machine vision system for UAV navigation; IEEE, 2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles and International Transportation Electrification Conference, ESARS-ITEC, 2016; pp.1–6. DOI: 10.1109/ESARS-ITEC.2016.7841356.
28. M. Ivanov, O. Sergiyenko, V. Tyrsa, P. Mercorelli, V. Kartashov, W. Hernandez, S. Sheiko, M. Kolendovska. Individual scans fusion in virtual knowledge base for navigation of mobile robotic group with 3D TVS // Proceedings of 44th Annual Conference of IEEE Industrial Electronics Society (IECON).. -2018. – Washington DC, USA. -S. 3187-3192. . ISBN 978-1-5090-6683-4/18/.

29. Murrieta-Rico, F.N., Petranovskii, V., Galvan, D.H., Sergiyenko, O., Yocupicio-Gaxiola, R.I., De Dios Sanchez-Lopez, J. Phase effect in frequency measurements of a quartz crystal using the pulse coincidence principle. 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 185-190, 9152255, DOI: 10.1109/ISIE45063.2020.9152255
30. Oleksandr Sotnikov, Vladimir Kartashov, Oleksandr Tymochko, Oleg Sergiyenko, Vera Tyrsa, Paolo Mercorelli, Wendy Flores-Fuentes. Methods for Ensuring the Accuracy of Radiometric and Optoelectronic Navigation Systems of Flying Robots in a Developed Infrastructure. Chapter 16// Machine Vision and Navigation; Springer, Cham. pp.537–578. Editors: Sergiyenko, Oleg, Flores-Fuentes, Wendy, Mercorelli, Paolo. DOI: 10.1007/978-3-030-22587-2_16.
31. Optical detection of unmanned air vehicles on a video stream in a real-time/Kartashov, V., Oleynikov, V., Zubkov, O., Sheiko, S.// 2019 International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019 - Proceedings, 2019, 9165362/
32. Principles Of Construction And Assessment Of Technical Characteristics Of Multi-Frequency Atmospheric Sodar In The Humidity Measurement Mode / Kartashov, V.M., Sidorov, G.I., Sheiko, S.A., Kolendovskaya, M.M., Sergienko, O.Yu. // Telecommunications And Radio Engineering (English Translation Of Elektrosvyaz And Radiotekhnika), 2020, ISSN Print: 0040-2508, ISSN Online: 1943-6009, DOI: 10.1615/TelecomRadEng.v79.i4.50, p. 323-333/
33. Research Of The Uncertainty Of Measurement Frequencies And Definitions Of The Frequency Signal In The Waveguide With Respect To Power / Semenets, V.Zakharov, I. Serhiienko, M., Kartashov, V.M, , Kolendovska, M., Hernandez, W., Hipolito, J.I.N., , Tyrsa, V.// 45th Annual Conference of the IEEE Industrial Electronics Society, IECON 2019; Lisbon Congress CenterLisbon; Portugal; 14 October 2019 до 17 October 2019; CFP19IEC-

ART; Код 155980, Volume 2019-October, October 2019, № 8927203, Pages 4674-4679

34. Spatial-Temporal Processing Of Acoustic Signals Of Unmanned Aerial Vehicles /Kartashov V.M., Oleinikov V.N., Zubkov O.V., Sheiko S.A., Kolendovska M.M.// Telecommunications And Radio Engineering (English Translation Of Elektrosvyaz And Radiotekhnika), 2020, ISSN Print: 0040-2508, ISSN Online: 1943-6009, DOI: 10.1615/Telecomradeng.v79.i9.40, p. 769-780
35. Stereoscopic Vision Systems In Machine Vision, Models, And Applications (Book Chapter)/ Ramírez-Hernández, L.R., Rodríguez-Quiñonez, J.C., Castro-Toscano, M.J., Kolendovska, M., Murrieta-Rico, F.N.// Machine Vision And Navigation, 2019 Machine Vision and Navigation30 September 2019, Pages 241-265
36. StrelkovaT., KartashovV., Lytyuga A., Strelkov A. Theoretical Methods of Images Processing in Optoelectronic Systems. Chapter 16. // Biometrics: Concepts, Methodologies, Tools, and Applications; Oleg Sergiyenko and Julio C. Rodriguez-Quiñonez. (341p.), IGI Global, 2017; pp. 361-381. DOI: 10.4018/978-1-5225-0983-7.ch016.
37. StrelkovaT., KartashovV., Lytyuga A.,StrelkovA. Theoretical Methods of Images Processing in Optoelectronic Systems. Chapter 6// Developing and Applying Optoelectronics in Machine Vision; Oleg Sergiyenko and Julio C. Rodriguez-Quiñonez. (341p.) – USA, Herhey, IGI Global, 2016; pp.180-205.
38. Sytnik O., KartashovV. Methods and Algorithms for Technical Visionin Radar Introscopy. Chapter 13// Optoelectronics in Machine Vision-Based Theories and Applications. IGI Global, 2019; pp. 373-391.
39. The Use of Factorization and Multimode Parametric Spectra in Estimating Frequency and Spectral Parameters of Signal/Semenets, V., Kartashov, V., Sergiyenko, O., ...Rodriguez-Quinonez, J.C., Flores-Fuentes, W.//IEEE International Symposium on Industrial Electronics, 2020, 2020-June, p. 215-219

40. Unda, O.F., Hernandez, W., Vargas, O., Mendez, A., Sergiyenko, O., Tyrsa, V. Construction of a robotic platform of differential type for first-year students of electronic engineering, 2020 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, SPEEDAM 2020, 24-26 de junio de 2020, Sorrento, Italia, pp. 538-543, 9161870, DOI: 10.1109/SPEEDAM48782.2020.9161870
41. Use of Acoustic Signature for Detection, Recognition and Direction Finding of Small Unmanned Aerial Vehicles/Kartashov, V., Oleynikov, V., Koryttsev, I., ...Babkin, S., Selieznov, I.//Proceedings - 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2020, 2020, p. 377-380/
42. V. Semenets; Vladimir Kartashov ; Oleg Sergiyenko; Vyacheslav Tikhonov ; Paolo Mercorelli ; Sergiy Sheiko ; Nataliya Chmelarova. "The Use of Factorization and Multimode Parametric Spectra in Estimating Frequency and Spectral Parameters of Signal," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 215-219, doi: 10.1109/ISIE45063.2020.9152238.
https://ieeexplore.ieee.org/abstract/document/9152238
43. Wilmar Hernandez ; Alfredo Mendez ; Omar Flor-Unda ; Vicente Gonzalez-Posada ; Jose Luis Jimenez ; Oleg Sergiyenko ; Julio C. Rodriguez-Quiñonez ; Mykhailo Ivanov ; Ivan Menes Camejo ; Marina Kolendovska. "A comparative example between the use of PCA and MDS for image classification," 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, Netherlands, 17-19 of June 2020, pp. 1353-1358, doi: 10.1109/ISIE45063.2020.9152565.
https://ieeexplore.ieee.org/abstract/document/9152565
44. Карташов В.М. и др. Обработка сигналов в радиоэлектронных системах дистанционного мониторинга атмосферы; Харьков: ХНУРЭ, 2014. 312 с.

45. Карташов В.М., Олейников В.Н., Колендовская М.М., Тимошенко Л.П., Капуста А.И., Рыбников Н.В. Комплексирование изображений при обнаружении беспилотных летательных аппаратов// Радиотехника. (Харьков). 2020. Вып. 201; С.120-129.
46. Карташов В.М., Посошенко В.А., Цехмистро Р.И., Тимошенко Л.П., Колендовская М.М. Методы ориентации, навигации и контроля мобильных робототехнических платформ// Радиотехника. (Харьков). 2019. Вып. 199. С. 38-44.
47. Ситнік О.В., Карташов В.М. Радіотехнічні системи. Навч. посібник. Х.: Сміт, 2009. 448 с.