НАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено:
В.о. завідувача кафедри

_____ Микола ГРАЙВОРОНСЬКИЙ

“___”_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Математичні методи моделювання, розпізнавання образів та безпеки даних»
спеціальності: 113 «Прикладна математика»

на тему: Аналіз алгоритмів навчання з підкріпленням для моделювання поведінки безпілотних автомобілів

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи __ФІ-72_____
(шифр групи)

__Попригo Ярослав Леонідович_
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доцент кафедри інформаційної безпеки Родіонов Андрій Миколайович
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Рецензент
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 113 «Прикладна математика»
Освітньо-професійна програма «Математичні методи моделювання, розпізнавання образів та безпеки даних»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

_____
Микола ГРАЙВОРОНСЬКИЙ

"___"_____2021 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Попригo Ярослава Леонідовича
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз ефективності алгоритмів навчання з підкріпленням, за допомогою яких моделюється поведінка безпілотних автомобілів _____, науковий керівник роботи к. т. н. Андрій Миколайович Родіонов _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «_____» 2021 р. № _____

2. Термін подання студентом роботи червня 2021 р.

3. Вихідні дані до роботи існуючі алгоритми навчання з підкріпленням, програмне серидовище PyCharm, мова програмування python та бібліотеки

4. Зміст роботи Розроблення та візуалізація середовища для моделювання поведінки безпілотних автомобілів, порівняльний аналіз ефективності алгоритмів навчання з підкріпленням

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація, тези конференції

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Визначення з темою роботи	1.09.2020-30.09.2020	
2	Актуальність, область застосування, постановка задачі	30.09.2020-04.12.20	
3	Огляд існуючих рішень	05.02.2021-17.03.2021	
4	Аналіз роботи відомих алгоритмів	17.03.2021-12.04.2021	
5	Реалізація серидовища	12.04.2021-14.05.2021	
6	Представлення до захисту		

Здобувач вищої освіти _____ Ярослав ПОПРИГО
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Андрій РОДІОНОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота містить 40 сторінок, 10 ілюстрацій, і 12 джерел літератури.

Наразі питання автономного управлінням автомобілем стає все гострішим. Кожного дня багато компаній працюють над вдосконаленням багатьох чинників, щільно пов’язаних з поведінкою безпілотних автомобілів. Підвищення рівня безпеки пасажирів та інших учасників руху, усунення заторів, зниження часу подорожі, збільшення пропускної спроможності, збільшення економії палива – є одними з найголовніших завдань при моделюванні поведінки безпілотних автомобілів.

Розгляд та аналіз ефективності алгоритмів навчання, які моделюють цю поведінку, мають неопосередкований влив на вибір найбільш ефективного алгоритму навчання, що є однією з найактуальніших задач у цій галузі.

Для досягнення мети було використано:

- Основні алгоритми навчання з підкріпленням.
- Carla для моделювання середовища.
- Pycharm та python libs для реалізації алгоритмів.

Ключові слова: НАВЧАННЯ З ПІДКРІПЛЕННЯМ, МАШИННЕ НАВЧАННЯ, АЛГОРИТМИ, БЕЗПІЛОТНІ АВТОМОБІЛІ, CARLA , Q – LEARNING, SARSA, EXPECTED – SARSA

ABSTRACT

Graduate work contains 40 pages, 10 illustrations, and 12 sources of literature. One of the problems of computer vision is the background recovery.

Nowadays the issue of autonomous driving is becoming more acute. Every day, many companies are working to improve many factors closely related to the behavior of unmanned vehicles. Improving the level of safety of passengers and other road users, eliminating congestion, reducing travel time, reducing the capacity, increasing fuel economy - some of the most important plants.

Analysis of the effectiveness of learning algorithms that simulate this behavior provide a direct impact on the choice of the most effective learning algorithm, which is one of the most pressing problems in the field.

To achieve this goal were used:

- Basic reinforcement learning algorithms.
- Carla for modeling the environment.
- Pycharm and python libs to implement algorithms.

Keywords: REINFORCEMENT EXERCISES, MACHINE LEARNING, ALGORITHMS, SELF – DRIVING CARS, CARLA, Q - LEARNING, SARSA, EXPECTED - SARSA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RL (reinforcement learning) – навчання з підкріпленням,

Action – дія, яку виконує модель автомобіля,

Environment – серидовище, в якому моделується поведінка автомобіля,

Goal – кінцева точка, якої повинна досягти модель автомобіля,

Obstacle – перешкода (пішоход),

Position – позиція моделі автомобіля відносно width,

Reward – нагорода,

State – положення об’єкта в серидовищі,

Velocity – швидкість моделі автомобіля.

ВСТУП

Великі затори на дорогах є важливою цивілізаційною та комерційною проблемою, особливо в міських, густо населених районах. Це спричиняє затримки у дорозі, стрес водіїв, шум, проблеми з організацією громадського транспорту та об'їздів, велике забруднення повітря, споживання палива та енергії тощо.

Ще однією важливою проблемою, пов'язаною з автомобільним рухом (не лише великими заторами), є автомобільні аварії та їх наслідки: загибель людей пасажирів, пошкодження автомобілів тощо. За даними нацполіції, у 2020 році в Україні сталося 168 тис. ДТП. Як наслідок, було травмовано 32 тис. осіб, більше 3,5 тис. загинули.

Подібна ситуація і в інщих країнах. За даними ВОЗ, в 2018 році внаслідок ДТП загинуло 1,35 млн осіб. Тому виникає потреба в інноваційних рішеннях, здатних зменшити затори на дорогах, а також кількість та наслідки аварій. За даними Google (2015), близько 94% усіх автомобільних аварій пов’язані з людськими помилками, тому усунення цього фактора представляється найкращим способом зменшити ризик зіткнень.

Інноваційним рішенням для цього є розробка безпілотних автомобілів, здатних керувати автомобілем без будь-яких дій людини.

Мета і завдання дослідження

Багато компаній з автомобільної та IT-галузі намагаються створити власні робочі моделі таких автомобілів. Проте проведення реальних досліджень в області безпілотних автомобілів вимагають великої кількості ресурсів і як наслідок коштують дуже дорого.

Тому завданням дослідження було розробити модельне середовище і на простих моделях показати, як буде себе поводити автомобіль на реальній дорозі.

Наукова новизна отриманих результатів

Розроблено модельне серидовище, в якому змодельовані основні учасники дорожнього руху, основні елементи дорожньої системи та проведено ряд експериментів з метою виявлення найефективніших алгоритмів.

Практичне значення отриманих результатів

Практичне значення роботи полягає в тому, що отримані результати задачі можуть бути застосовними для проведення досліджень з безпілотними автомобілями у реальному середовищі.

Апробація результатів роботи

Робота була оприлюднена у вигляді доповіді на XIX Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених “Теоретичні і прикладні проблеми фізики, математики та інформатики”.

Публікації

Робота була опублікована на XIX Всеукраїнській науково- практичній конференції студентів, аспірантів та молодих вчених “Теоретичні і прикладні проблеми фізики, математики та інформатики”.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ МОДЕЛЮВАННЯ ПОВЕДІНКИ БЕЗПІЛОТНИХ АВТОМОБІЛІВ

У цьому розділі буде проаналізовано вже існуючі підходи до моделювання поведінки безпілотних автомобілів.

Моделювання дорожнього руху є важливим аспектом транспортних досліджень. Його метою є розробка математичних інструментів, що описують реальний трафік із бажаною точністю. Існують різні підходи до моделювання дорожнього руху з урахуванням рівня деталізації: від мікроскопічного рівня (кожен автомобіль змодельований як окремий агент, модель описує еволюцію стану автомобіля, наприклад, положення та швидкість) до макроскопічного (рух, що описується співвідношеннями між агрегованими значеннями , наприклад, швидкість, витрата).

Далі буде розглянута мікроскопічна імітаційна модель для безпілотних автомобілів.

1.1 Опис імітаційної моделі для безпілотних автомобілів

Представлення руху, розроблене з метою дослідження автономних автомобілів, належить до групи мікроскопічних моделей руху. Це означає, що на кожному етапі моделювання кожен транспортний засіб розглядається як окремий об'єкт, а динамічні параметри транспортного засобу, наприклад, його положення та швидкість, оновлюються локально в часі.

Що стосується руху транспортних засобів, було застосовано два основних спрощення. По-перше, розглядається лише сітка перпендикулярних смуг для руху. По-друге, поворот імітується як процес, який складається з наступних етапів:

- набуття швидкості, яка дозволяє повертати (у більшості випадків просто зменшення швидкості).

- потрапляння в положення, коли дві смуги перетинаються одна з одною.
- миттєва зміна напрямку.
- рух у новому напрямку.

В результаті транспортний засіб може рухатись лише у чотирьох основних напрямках (N, E, S, W), при цьому він завжди повертає під кутом 90 градусів. Окрім смуг, які призначені за фіксованими напрямками, ще одним елементом інфраструктури, що розглядається, є світлофор. Транспортні засоби отримують інформацію (I2V) про поточну фазу світлофора (зупинка / продовження) та час, який пройде до закінчення поточної фази [12].

1.2 Представлення агентів

Спосіб моделювання дорожнього руху представлений у вигляді багатоагентної системи: кожен транспортний засіб є індивідуальним агентом в рамках моделювання.

Відповідно до концепції агента „Переконання-бажання-наміри” (BDI, Rao та Georgeff (1991)), основною метою агента є задоволення його бажань шляхом досягнення деяких чітко визначених цілей. Для планування та виконання дій, що ведуть до досягнення встановлених цілей, агенту необхідно отримати знання про навколишнє середовище. Вся інформація про інших агентів, перешкоди тощо формує впевненість агента у його діях.

Припускається, що інформація про навколишнє середовище (наприклад, карта, світлофор, положення перешкод у певному діапазоні) є абсолютно доступною для автономних автомобілів за допомогою датчиків (наприклад, радарів, лідарів, відеокамер) та зв'язку I2V (infrastructure-to-vehicle – інфраструктура до автомобіля). Екземпляр симулятора подає агентам дані про все, що існує або відбувається в межах визначеного діапазону, і не залучає інших автономних автомобілів. Агенти отримують інформацію про інші автономні

машини в межах визначеного діапазону за допомогою зв'язку V2V (vehicle-to-vehicle – автомобіль до автомобіля). Ці два типи отримання даних моделюються окремо, щоб імітувати реальні умови дорожнього руху, в яких знання про інші автономні транспортні засоби будуть отримуватися переважно за допомогою зв'язку V2V. Сучасна конструкція передбачає, що отримання даних триває незначно короткий проміжок часу [12].

1.3 Дотримання вимог

Дотримання вимог представляють правила, яких транспортний засіб дотримується, та загальні вимоги, які він зобов'язаний виконувати (наприклад, досягнення пункту призначення, запобігання зіткненню). Зручно запровадити механізм, який отримує необхідну інформацію (переконання) та планує дії транспортного засобу, щоб все, що робить транспортний засіб, відповідало його бажанням. Вводиться також властивість бажання, пов’язана з його значимістю. Оскільки деякі бажання мають вирішальне значення для безпеки транспортного засобу, а деякі просто висловлюють занепокоєння з приводу комфорту їзди, зручно оперувати ієрархіями бажань. Бажання слідувати маршруту та воля уникати аварій (що важливіше) служать прикладом основних бажань.

1.4 Мета та цілі

Передбачається, що кожен агент (автономний автомобіль) має маршрут, призначений на самому початку своєї подорожі. Спираючись на маршрут, агент може визначити цілі, які він має намір досягти. Одна ціль описує необхідний стан транспортного засобу, який складається з усіх динамічних даних транспортного засобу. Простір станів містить поля з інформацією про:

- позицію (пара координат: (x_1, x_2)).
- швидкість (значення, швидкість та напрямок).

- прискорення (значення та напрямок).

Наприклад, якщо транспортний засіб їде на схід і повинен повернути праворуч, коли прибуде у (або близько до) положення x, y, де x ∈ $[x_1, x_2]$ (діапазон, в межах якого транспортним засобам дозволено повертати на розв'язку), встановлюються дві цілі:

Таблиця 1.1 – Вибір цілі в залежності від швидкості

State	Goal №1	Goal №2
Position	x ∈ $[x_1, x_2]$	x ∈ $[x_1, x_2]$
Velocity speed (v)	v ∈ $[v_1, v_2]$	Undefined
Velocity direction (N, E, S, V)	E	S

При цьому вважається, що v_1 і v_2 – це мінімальна та максимальна швидкості відповідно, які дозволені та придатні для повороту. Їх значення визначаються для кожного перехрестя [12].

1.5 Дії

Одинарна дія - це механізм, який змінює стан транспортного засобу протягом одного кроку. Може знадобитися, щоб транспортний засіб відповідав конкретним вимогам до початку виконання. Це служить для імітації реальних ситуацій, коли транспортний засіб не може виконати певний маневр, якщо його значення швидкості не включено в заданий діапазон.

1.6 Крок моделювання

Процес моделювання ділиться на етапи. Виконання кожного кроку передбачає:

- Зв'язок з середовищем– транспортні засоби отримують інформацію про середовище.
- Планування – планування дій, які слід виконати на наступному етапі.
- Виконання – дії, заплановані на поточному етапі, транспортні засоби рухаються вперед.

Інтервал часу між кроками гнучко регулюється. Якщо його значення невелике, знання транспортного засобу про стан навколишнього середовища часто оновлюються. Це призводить до того, що транспортний засіб може легко регулювати свою поведінку в динамічних ситуаціях. Недоліком, який слід врахувати, є те, що процес взаємодії та планування може також зайняти багато часу. З іншого боку, якщо інтервал часу довгий, транспортні засоби повинні бути обережнішими при плануванні своїх дій (їх не можна змінити до наступного кроку). Як наслідок, деякі можливості для прискорення втрачаються, і їзда виявляється не оптимальною.

1.7 Зв'язок з серидовищем

На цьому етапі транспортний засіб отримує знання про навколишнє середовище на фіксованій відстані. Інформація про інфраструктуру (включаючи наявність та фазу світлофора) надається екземпляром симулятора. Кожен транспортний засіб також отримує повідомлення від сусідніх автомобілів із даними про їх поточний стан (транспортні засоби, що спереду, також повідомляють про їх плани на поточний крок). При цьому він зобов'язаний транслювати повідомлення, що містить власні параметри та визначені дії. У разі потреби виникнення необхідності домовлятися про пріоритети або планувати альтернативні маршрути разом. У поточній конструкції було зроблено припущення, що зв'язок між агентами є абсолютно надійним. Час, необхідний для зв'язку між транспортними засобами, враховується приблизно. Протягом

інтервалу кроків транспортним засобом може бути надіслано або прийнято лише обмежену кількість повідомлень. Кількість повідомлень, якими обмінюються в одній розмові, також обмежена часом.

1.8 Планування

Одразу після того, як переконання агента оновлені, потрібно скласти план принаймні поточного кроку. Для прийняття правильних рішень та підготовки правильних дій транспортний засіб використовує свою базу знань та бажання. На початку створюється нейтральна дія. Потім воно фільтрується бажаннями на всіх рівнях.

Процес починається із модифікацій початкової дії. Їх мета - збільшити швидкість, щоб вона досягла бажаного значення, яке встановлюється фіксованим для кожного автомобіля. Потім модифікована дія відкладається.

На наступному рівні транспортний агент перевіряє, чи є якісь цілі для досягнення. Якщо так, то створюються нові цільові дії на поточному етапі. В іншому випадку дія, створена бажанням їхати швидко, є тією, яку слід взяти до уваги.

Останній етап передбачає необхідні зміни в плані дій, щоб транспортний засіб не врізався в транспортні засоби спереду. Для виявлення та уникнення небезпечних ситуацій формулюється стан безпеки.

Розглянемо транспортний засіб, що рухається вздовж осі x зі швидкістю $v = [v_x, 0]$. Його максимальне уповільнення дорівнює a_x.

Якщо транспортний засіб знаходиться в положенні (x_0, y), гальмуючи з максимальним уповільненням, він зупиниться в положенні (x_s, y), де:

$$x_s = x_0 + \frac{v_x^2}{2a_x} \tag{1.1}$$

Якщо сусід автомобіля в межах діапазону і перед вибраним транспортним засобом в даний час знаходиться в положенні (x_0', y) подорожі зі швидкістю $v' = [v_x', 0]$ мають довжину d і максимальне уповільнення a_x', має бути виконана

наступна умова:

$$x_s' - x_s > d \tag{1.2}$$

де (x_s', y) позиція сусіда після його уповільнення з a_x' і зупинки.

Еквівалентно, використовуючи формулу (1):

$$x_s' - x_s = x_0' + \frac{v_x'}{2a_x'} - x_0 - \frac{v_x^2}{2a_x} > d \tag{1.3}$$

Інтуїтивно, якщо виконується умова безпеки, тоді: якщо будь-який транспортний засіб спереду та в діапазоні оригінального почне гальмувати, є гарантія, що вибраний транспортний засіб зможе швидко уповільнитись, і завжди буде не нульова відстань між цими двома автомобілями. На даний момент роль "бажання безпеки" зводиться до вирішення питання, чи підтримувати швидкість транспортного засобу результатом попередніх бажань (так, якщо ситуація, що виникає, задовольняє (3)), або швидкість слід негайно зменшити (якщо поточна ситуація порушує (3)) [12].

1.9 Виконання

Як тільки процес планування закінчується для кожного з модельованих транспортних засобів, всі дії, заплановані на поточному етапі, виконуються по одному. Потім перевіряється стан нового транспортного засобу з метою виявлення аварій.

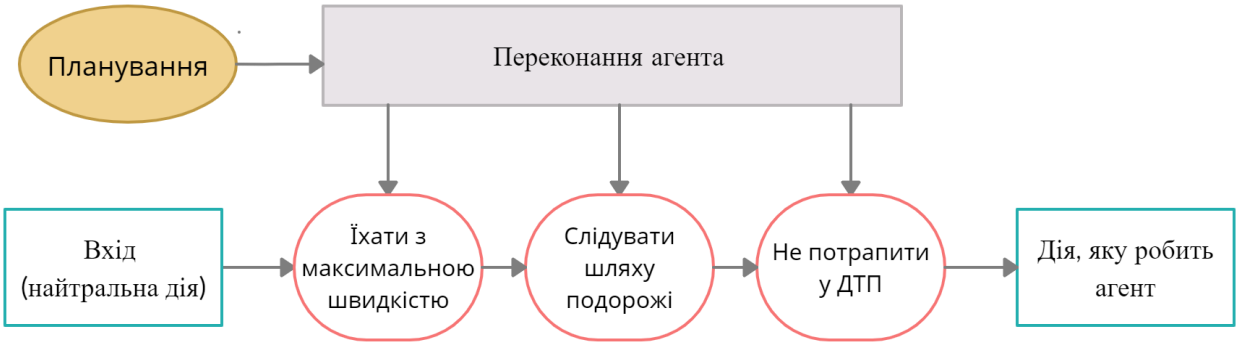

Рисунок 1.1 – Процес прийняття рішення щодо дії агентом

Висновки до розділу 1

У цьому розділі було роглянуто існуючі підходи до моделювання поведінки безпілотних автомобілів, а також були описані: імітаційна модель автомобіля, його простір дій, представлення агентів. Були описані мета та цілі агента, визначені вимоги, яких він має дотримуватись. В результаті було предсталено підхід до моделювання, який складається з трьох етапів: зв'язок з серидовищем, планування та виконання, в яких детально описано основні форми поведінки агента в серидовищі.

2 АНАЛІЗ РОБОТИ ВІДОМИХ АЛГОРИТМІВ

У цьому розділі буде поставлена задача навчання з підкріпленням та описані алгоритми, застосовні до задачі моделювання поведінки безпілотних автомобілей.

2.1 Постановка задачі навчаня з підкріпленням

Навчання з підкріпленням моделюється як Марковський процес прийняття рішення (МППР), в якому:

- S – множина станів середовища,
- A – сукупність дій агента,
- $P_a(s, s') = Pr(s_{t+1} = s' | s_{t+1} = s,\ a_t = a)$ – ймовірність переходу від стану s до стану s' під дією a в момент часу t .
- $R_a(s, s')$ - нагорода після переходу з стану s до стану s' під дією a.

Алгоритм взаємодії агента з середовищем:

1. Ініціалізуємо стратегію $\pi_1(a|s)$ та стани середовища s_1
2. Для будь- якого $t = 1, \dots T, \dots$
3. Агент вибирає дію $a_t \sim \pi_t(a|s_t)$;
4. Середовище генерує винагороду $r_{t+1} \sim p(r \mid a_t, s_t)$

   Та новий стан $s_{t+1} \sim p(r \mid a_t, s_t)$;
5. Агент вносить зміни до стратегії $\pi_{t+1}(a|s)$;

При цьому, $R_t = r_{t+1} + r_{t+2} + \dots + r_{t+k} + \dots$ – сумарна винагорода, а в загальному випадку $R_t = r_{t+1} + \gamma \cdot r_{t+2} + \dots + \gamma^{k-1} \cdot r_{t+k} + \dots$,

де $\gamma \in [0, 1]$ – коефіцієнт знижки – чим вище значення γ, тим більше агент дальньовидний [10][3].

Типовий вигляд сценарію RL: агент здійснює дії в серидовищі, що інтерпретується в винагороду та представлення стану, що передаються назад агентові.

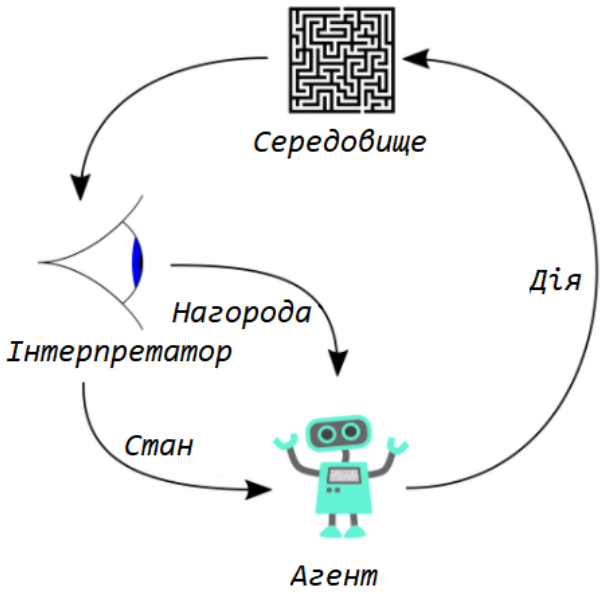

Рисунок 2.1 – Вигляд сценарію RL

Мета усього навчання для підкріплення - максимізувати суму майбутніх винагород. [8] [2]

Існує 2 основних типи алгоритмів RL. Вони засновані на моделях (model-based) та без моделей (model-free). Далі буде розглянуто алгоритми, засновані на концепції model-free [4].

2.2 Опис алгоритму Q – learning

Перед описом алгоритму доцільно розглянути основну концепцію Q – learning, яка задається рівнянням Беллмана:

$$Q(s, a) = r + \gamma max_{a'} Q(s', a')$$

Рівняння стверджує, що значення Q для певної пари стан-дія повинно бути винагородою, отриманої при переході в новий стан (шляхом виконання цієї дії), доданої до значення найкращої дії в наступному стані. Іншими словами, ми поширюємо інформацію про значення дій по одному кроку за раз.

Проте ми можемо вирішити, що отримання нагороди прямо зараз цінніше, ніж отримання нагороди в майбутньому, і тому у нас є γ, число від 0 до 1 (зазвичай від 0,9 до 0,99), яке множиться на нагороду в майбутньому, знецінюючи майбутні нагороди. [3]

Опис алгоритму [9]:

Функція навчання: $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

1. Множина станів $\mathcal{X} = \{1, \dots, n_x\}$
2. Множина дій $\mathcal{A} = \{1, \dots, n_a\}$, $A: \mathcal{X} \Rightarrow \mathcal{A}$
3. Функція винагород: $R: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
4. Імовірнісна функція переходу $T: \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
5. Швидкість навчання $\alpha \in [0,1]$, зазвичай $\alpha = 0.1$
6. Коефіцієнт знижки $\gamma \in [0,1]$

Procedure $QLearning\ (\mathcal{X}, \mathcal{A}, R, T)$

Довільно ініціалізуємо $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

While Q не збігається **do**

Починаємо зі стану $s \in \mathcal{X}$

While s не кінцеве **do**

Підраховуємо π відповідно до Q по стратегії розвідки (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)

$a \leftarrow \pi(s)$

$r \leftarrow R(s, a)$ ► Отримуємо винагороду

$s' \leftarrow T(s, a)$ ►Отримуємо новий стан

$Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$

$s \leftarrow s'$

Return Q

2.3 Опис алгоритму SARSA

Опис алгоритму [9]:

Функція навчання: $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

1. Множина станів $\mathcal{X} = \{1, \dots, n_x\}$
2. Множина дій $\mathcal{A} = \{1, \dots, n_a\}$, $A: \mathcal{X} \Rightarrow \mathcal{A}$
3. Функція винагород: $R: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
4. Імовірнісна функція переходу $T: \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
5. Швидкість навчання $\alpha \in [0,1]$, зазвичай $\alpha = 0.1$
6. $\lambda \in [0;1]$
7. Коефіцієнт знижки $\gamma \in [0,1]$

Procedure $SARSA\ (\mathcal{X}, \mathcal{A}, R, T, \alpha,\ \gamma,\ \lambda)$

Довільно ініціалізуємо $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

While Q не збігається **do**

Вибираємо $(s, a) \in \mathcal{X} \times \mathcal{A}$

While s не кінцеве **do**

$r \leftarrow R(s, a)$ ► Отримуємо винагороду

$s' \leftarrow T(s, a)$ ►Отримуємо новий стан

Підраховуємо π відповідно до Q (наприклад по стратегії *epsilon-greedy*)

$a' \leftarrow \pi(s')$

$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma Q(s', a'))$

$s \leftarrow s'$

$a \leftarrow a'$

Return Q

Якщо стан S є кінцевим (стан цілі або кінцевий стан), то Q (S, a) = 0 ∀ a ∈ A, де A - сукупність усіх можливих дій.[3]

2.4 Опис алгоритму Expected – SARSA

Опис алгоритму [9]:

Функція навчання: $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

1. Множина станів $\mathcal{X} = \{1, \dots, n_x\}$
2. Множина дій $\mathcal{A} = \{1, \dots, n_a\}$, $A: \mathcal{X} \Rightarrow \mathcal{A}$
3. Функція винагород: $R: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
4. Імовірнісна функція переходу $T: \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
5. Швидкість навчання $\alpha \in [0,1]$, зазвичай $\alpha = 0.1$
6. $\lambda \in [0;1]$
7. Коефіцієнт знижки $\gamma \in [0,1]$

Procedure $SARSA\ (\mathcal{X}, \mathcal{A}, R, T, \alpha,\ \gamma,\ \lambda)$

Довільно ініціалізуємо $Q: \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

While Q не збігається **do**

Вибираємо $(s, a) \in \mathcal{X} \times \mathcal{A}$

While s не кінцеве **do**

$r \leftarrow R(s, a)$ ► Отримуємо винагороду

$s' \leftarrow T(s, a)$ ►Отримуємо новий стан

Підраховуємо π відповідно до Q (наприклад по стратегії *epsilon-greedy*)

$a' \leftarrow \pi(s')$

$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \sum_a \pi(a|s') \cdot Q(s', a'))$

$s \leftarrow s'$

$a \leftarrow a'$

Return Q

Порівняння функцій дія – значення (action - value):

- **SARSA:**
$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$
- **Q-Learning:**
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$
- **Expected SARSA:**
$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \sum_a \pi(a|s_{t+1}) Q(s_{t+1}, a) - Q(s_t, a_t))$$

Рисунок 2.2 – Порівняння функцій дія – значення

Ми бачимо, що Expected – SARSA приймає зважену суму всіх можливих наступних дій з урахуванням ймовірності вчинення поточної дії. Q – learning, на відміну від SARSA, коли дію a було обрано, дотримуючись певної політики, то дію a' вибирає, просто взявши за неї максимум Q [6].

Висновки до розділу 2

У цьому розділі була поставлена задача навчання з підкріпленням, введено математичне підґрунтя до використання базових алгоритмів навчання, а також детально описано алгоритми Q – learning, Sarsa та Expected – Sarsa.

3 ПОРІВНЯННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ НАВЧАНННЯ З ПІДКРІПЛЕННЯМ

У цьому розділі побудовано просте серидовище, за допомогою якого розглянуто ефективність роботи алгоритмів навчання з підкріпленням для конкретного прикладу використання.

3.1 Постановка задачі

Маємо: модель автомобіля, перешкоду у вигляді пішохода, кінцеву точку. Завдання моделі автомобіля досягти кінцевої точки, при цьому зменшити швидкість, проїзджаючи повз пішохода, затративши на дорогу якомога менше часу.

Рисунок 3.1 – Візуалізація дискретного середовища

Більш детальне представлення середовища зображено на рисунку Б.1

3.2 Параметри моделі

Перелік можливих дій моделі автомобіля Actions:

- $no\ change$ – підтримування поточної швидкості
- $speed\ up$ – підвищення швидкості
- $hard\ speed\ up$ – сильне підвищення швидкості
- $slow\ down$ – уповільнення.
- $hard\ slow\ down$ – сильне уповільнення.

Визначення функції reward:

- проїзд цілі з $velocity = 30 : 40$
- проїзд цілі з $velocity\ != 30 : -40$.
- $action\ change: -3$.
- resources: -1.
- проїзд повз перешкоду з *very_bad_speed_near_pedestrian*: -40.
- проїзд повз перешкоду з *bad_speed_near_pedestrian*: -30.
- проїзд повз перешкоду з *not_so_bad_speed_near_pedestrian*: -20.

Особливості стану (state features):

- position.
- velocity.

Координати кінцевої точки:

- {19, 1}. [1]

3.3 Опис алгоритмів

3.3.1 Q – навчання

Завдання алгоритму - оновити q-таблицю на основі спостережуваного досвіду S.A.R.S.

Параметри алгоритму:

- s: попередній стан.
- a: поточна дія.
- r: нагорода.
- s*: наступний стан.
- gamma: коеф. знижки.
- lr: - швидкість навчання.

В цьому випадку очікуване значення (q*) - це максимальне значення відфільтрованого стовпця q- таблиці, а значення target задається у вигляді виразу:

$$r + gamma \times q*$$

3.3.2 Sarsa

Завдання алгоритму - оновити q-таблицю на основі спостережуваного досвіду S.A.R.S.A використовую- чи поточну дію a, для оцінки Q (s*, a*).

Параметри алгоритму:

- s: попередній стан.
- a: поточна дія.
- r: нагорода.

- s*: наступний стан.
- gamma: коеф. знижки.
- lr: - швидкість навчання.

В цьому випадку очікуване значення (q*) – це q – значення пари (s*, a*), а значення target задається у вигляді виразу:

$$r + gamma \times q*$$

3.3.3 Expected - Sarsa

Завдання алгоритму - оновити q-таблицю на основі спостережуваного досвіду S.A.R.S.A використовуючи очікуване q – значення наступного стану q* для побудови q_target , для оцінки Q (s*, a*).

Параметри алгоритму:

- s: попередній стан.
- a: поточна дія.
- r: нагорода.
- greedy_epsilon: "жадібний епсилон", значення від 0 до 1.
- gamma: коеф. знижки.
- lr: - швидкість навчання.

В цьому випадку очікуване значення (q*) рахується за формулою (1 - greedy_epsilon) * q_max + greedy_epsilon * q_mean, а значення target задається у вигляді виразу:

$$r + gamma \times q*$$

3.4 Результати

У результаті роботи алгоритмів були побудовані такі графіки:

Рисунок 3.2 – Результат роботи алгоритму Q – learning

Рисунок 3.3 – Результат роботи алгоритму Sarsa

Рисунок 3.4 – Результат роботи алгоритму Expected – Sarsa

Та побудований порівняльний графік ефективності цих алгоритмів:

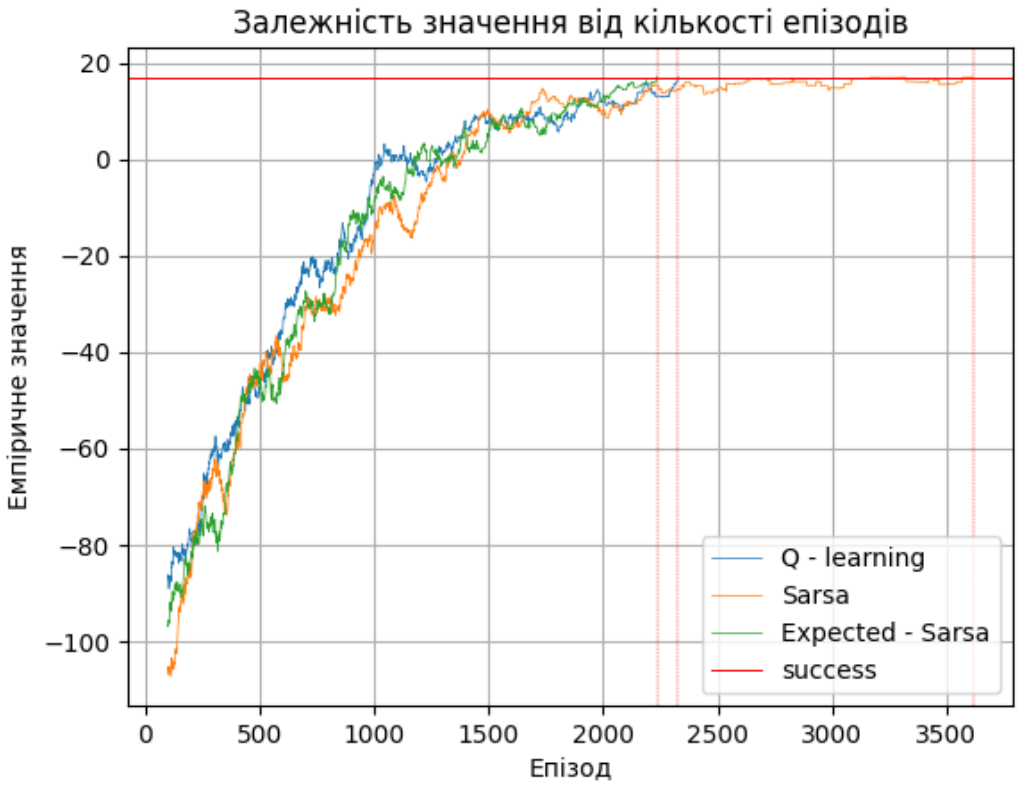

Рисунок 3.5 – Порівняльний графік ефективності алгоритмів

Тож бачимо, що у результаті навчання алгоритмів Q – learning, Sarsa, та Expected – Sarsa, вдалося визначити найефективніший алгоритм навчання з підкріпленням застосовний до розробленого сердивища – алгоритм Expected – Sarsa. Алгоритм Q – learning навчився моделювати поведінку моделі автомобіля за 2350 епізодів, в той час як алгоритм Expected – Sarsa навчився за 2200 епізодів, а алгоритм Sarsa - за 3700 епізодів.

Висновки до розділу 3

У цьому розділі було розроблене серидовище для моделювання поведінки безпілотних автомобілів та застосовані до моделі такі алгоритми навчання з підкріпленням: Q – learning, Sarsa, Expected – Sarsa. В результаті роботи алгоритмів було виявлено найбільш ефективний для визначеного прикладу

використання – алгоритм Expected – Sarsa, якому вдалось за 2200 епізодів досягти прийнятного емпіричного значення. Алгоритму Q – learning вдалося досягти цього значення за 2350 епізодів, а алгоритму Sarsa – за 3700 епізодів.

ВИСНОВКИ

В результаті роботи було:

1) Роглянуто існуючі підходи до моделювання поведінки безпілотних автомобілів, а також були описані: імітаційна модель автомобіля, його простір дій, представлення агентів. Були описані мета та цілі агента, визначені вимоги, яких він має дотримуватись. В результаті було предсталено підхід до моделювання, який складається з трьох етапів: зв'язок з серидовищем, планування та виконання, в яких детально описано основні форми поведінки агента в серидовищі.
2) Поставлена задача навчання з підкріпленням, введено математичне підґрунтя до використання базових алгоритмів навчання, а також детально описано алгоритми Q – learning, Sarsa та Sarsa – lambda.
3) Розроблене серидовище для моделювання поведінки безпілотних автомобілів та застосовані до моделі такі алгоритми навчання з підкріпленням: Q – learning, Sarsa, Expected – Sarsa.

У результаті роботи алгоритмів Q – learning, Sarsa, та Expected – Sarsa, вдалося визначити найефективніший алгоритм навчання з підкріпленням застосовний до розробленого сердивища – алгоритм Expected – Sarsa. Алгоритм Q – learning навчився моделювати поведінку моделі автомобіля за 2350 епізодів, в той час як алгоритм Expected – Sarsa навчився за 2200 епізодів, а алгоритм Sarsa - за 3700 епізодів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. chauvinsimon/reinforcement-learning-for-decisionmaking-in-self-driving-cars: Reinforcementlearning-for-decision-making-in-self-driving-cars.
2. Shweta Bhatt. Things you need to know about reinforcement learning. Retrieved July, 7:2019, 5.
3. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
4. Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In European Conference on Machine Learning, pages 317–328. Springer, 2005.
5. Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesv´ari. Convergence results for single-step on-policy reinforcement-learning algorithms. Machine learning, 38(3):287–308, 2000.
6. Reinforcement learning: Temporal-Difference, SARSA, Q-Learning & Expected SARSA in python . https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e.
7. RS Sutton and AG Barto. Reinforcement learning: An introduction. in progress, complete draft online, 2017.
8. 5 Things You Need to Know about Reinforcement Learning. https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html.
9. Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. https://digital-library.theiet.org/content/journals/10.1049/iet-its.2019.0317.
10. Обучение с подкреплением (Reinforcement Learning).

http://www.machinelearning.ru/wiki/images/archive/3/35/20140621071329%21Voron-ML-RL-slides.pdf.

11. Построить беспилотный автомобиль за месяц — инструкция от любителя. https://vc.ru/transport/24494-selfdriving-car-in-month.
12. Traffic Models for Self-driving Connected Cars. https://www.sciencedirect.com/science/article/pii/S2352146516302423.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Graphics.py:

```
import numpy as np
import matplotlib.pyplot as plt

returns_smoothed_1 = np.load(file='src/returns_smoothed_1.npy')
returns_smoothed_2 = np.load(file='src/returns_smoothed_2.npy')
returns_smoothed_3 = np.load(file='src/returns_smoothed_3.npy')
plt.figure()
plt.grid(True)
# plt.plot(returns_to_plot, linewidth=0.5)
plt.plot(returns_smoothed_1, linewidth=0.5, label='q')
plt.plot(returns_smoothed_2, linewidth=0.5, label='sarsa')
plt.plot(returns_smoothed_3, linewidth=0.5, label='sarsa-lambda')
plt.axhline(y=17, color='r', linewidth=0.7, linestyle='-', label='success')
for yc in [returns_smoothed_1, returns_smoothed_2, returns_smoothed_3]:
    plt.axvline(x=len(yc), linewidth=0.3, color='r', linestyle='--')
plt.xlabel("Episode")
plt.ylabel("Episode Return")
plt.title(f"Episode Return over Time")
plt.savefig("results/simple_road/analysis.png", dpi=800)
plt.legend()
plt.show()
```

Spawn_vehicles.py:

```
import glob
import os
import sys
try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass
import carla
```

```
import random
import time
def main():
    actor_list = []
    try:
        client = carla.Client(host='127.0.0.1', port=2000)
        client.set_timeout(2.0)
        world = client.get_world()
        blueprint_library = world.get_blueprint_library()
        bp = random.choice(blueprint_library.filter('vehicle'))
        if bp.has_attribute('color'):
            color = random.choice(bp.get_attribute('color').recommended_values)
            bp.set_attribute('color', color)
        transform = random.choice(world.get_map().get_spawn_points())
        # So let's tell the world to spawn the vehicle.
        vehicle = world.spawn_actor(bp, transform)
        actor_list.append(vehicle)
        print('created %s' % vehicle.type_id)
        vehicle.set_autopilot(True)
        camera_bp = blueprint_library.find('sensor.camera.depth')
        camera_transform = carla.Transform(carla.Location(x=1.5, z=2.4))
        camera      =      world.spawn_actor(camera_bp,      camera_transform,
attach_to=vehicle)
        actor_list.append(camera)
        print('created %s' % camera.type_id)
        cc = carla.ColorConverter.LogarithmicDepth
        camera.listen(lambda   image:   image.save_to_disk('_out/%06d.png'   %
image.frame_number, cc))
        location = vehicle.get_location()
        location.x += 40
        vehicle.set_location(location)
        print('moved vehicle to %s' % location)
        transform.location += carla.Location(x=40, y=-3.2)
        transform.rotation.yaw = -180.0
        for _ in range(0, 10):
            transform.location.x += 8.0
            bp = random.choice(blueprint_library.filter('vehicle'))
            npc = world.try_spawn_actor(bp, transform)
            if npc is not None:
```

```
                actor_list.append(npc)
                npc.set_autopilot()
                print('created %s' % npc.type_id)
        time.sleep(5)
    finally:

        print('destroying actors')
        for actor in actor_list:
            actor.destroy()
        print('done.')
if __name__ == '__main__':
    main()
```

Spawn_walkers.py:

```
import glob
import os
import sys
try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass
import carla
import random
import time
def main():
    actor_list = []
    try:
        client = carla.Client(host='127.0.0.1', port=2000)
        client.set_timeout(2.0)
        world = client.get_world()
        blueprint = random.choice(world.get_blueprint_library().filter('walker.*'))
        spawn_points = world.get_map().get_spawn_points()
        spawn_point = random.choice(spawn_points) if spawn_points else carla.Transform()
        world.try_spawn_actor(blueprint, spawn_point)
```

```
        time.sleep(10)
    finally:
        print('destroying actors')
        for actor in actor_list:
            actor.destroy()
        print('done.')
if __name__ == '__main__':
    main()
```

Algorithms.py:

```
import numpy as np
import pickle
from copy import copy
import pandas as pd
from matplotlib.pyplot import *
import random
from abc import ABC, abstractmethod
import os
from collections import defaultdict


class SarsaTable():
    def __init__(self, actions, state, load_q_table=False):
        super(SarsaTable, self).__init__(actions, state, load_q_table)
    def learn(self, s, a, r, s_, a_, termination_flag, gamma, learning_rate):
        self.check_state_exist(s_)
        id_row_previous_state = self.get_id_row_state(s)
        id_row_next_state = self.get_id_row_state(s_)
        q_predict = self.q_table.loc[id_row_previous_state, a]
        if termination_flag:
            q_target = r
        else:
            q_expected = self.q_table.loc[id_row_next_state, a_]
            q_target = r + gamma * q_expected
        self.q_table.loc[id_row_previous_state, a] += learning_rate * (q_target
- q_predict)
class ExpectedSarsa():
    def __init__(self, actions, state, load_q_table=False):
        super(ExpectedSarsa, self).__init__(actions, state, load_q_table)
```

```
    def learn(self, s, a, r, s_, termination_flag, greedy_epsilon, gamma,
learning_rate):
        self.check_state_exist(s_)
        id_row_previous_state = self.get_id_row_state(s)
        id_row_next_state = self.get_id_row_state(s_)
        q_predict = self.q_table.loc[id_row_previous_state, a]
        if termination_flag:
            q_target = r
        else:
            row = self.q_table.loc[id_row_next_state]
            filtered_row = row.filter(self.actions_list)
            q_max = max(filtered_row)
            q_mean = 0
            if len(filtered_row):
                q_mean = sum(filtered_row)/len(filtered_row)
            q_expected = (1 - greedy_epsilon) * q_max + greedy_epsilon * q_mean
                        q_target = r + gamma * q_expected
        self.q_table.loc[id_row_previous_state, a] += learning_rate * (q_target
- q_predict)
class QLearning():
    def __init__(self, actions, state, load_q_table=False):
        super(QLearningTable, self).__init__(actions, state, load_q_table)
    def learn(self, s, a, r, s_, termination_flag, gamma, learning_rate):
        self.check_state_exist(s_)
        id_row_previous_state = self.get_id_row_state(s)
        id_row_next_state = self.get_id_row_state(s_)
        q_predict = self.q_table.loc[id_row_previous_state, a]
        if termination_flag:
            q_target = r
        else:
            row = self.q_table.loc[id_row_next_state]
            filtered_row = row.filter(self.actions_list)
            q_expected = max(filtered_row)
            q_target = r + gamma * q_expected
        self.q_table.loc[id_row_previous_state, a] += learning_rate * (q_target
- q_predict)
```

ДОДАТОК Б ВЕЛИКІ РИСУНКИ

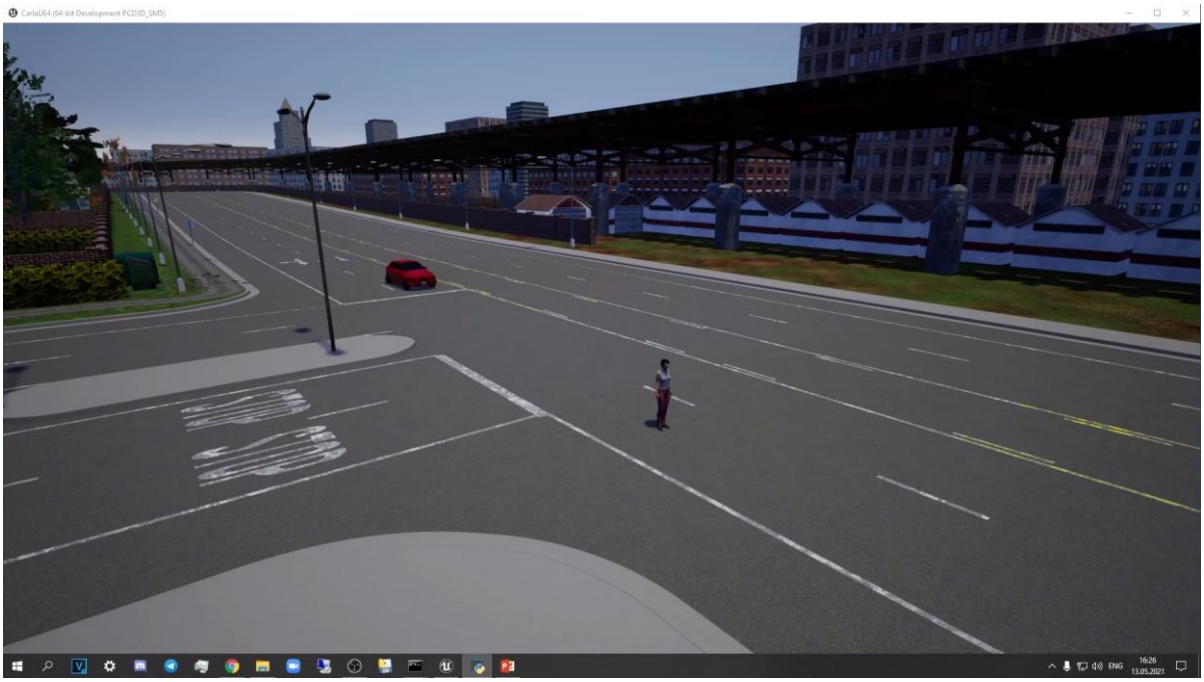

Рисунок Б.1 – Візуалізація середовища за допомогою Carla