DOI 10.34185/1562-9945-6-137-2021-13
УДК 519.689:004.42

Є.І. Булатецький, В.Г. Зайцев

ПРО ОДИН ПІДХІД ДО РОЗРОБКИ СИМУЛЯТОРУ РУХУ АВТОНОМНОГО ТРАНСПОРТНОГО ЗАСОБУ З НАВЧАННЯМ

Анотація. У статті розглядається питання використання тренажерів для управління рухом автономного транспортного засобу та розробки нового симулятору. Описано підхід до створення симулятору руху транспортного засобу на мові програмування C#. При розробки для реалізації симуляційних сцен, використано багатоплатформовий інструмент Unity 3D. У симуляції використовується нейронна мережа прямого поширення, яка немає чіткої кількості нейронів вхідного рівня, має тільки сталий вихідний рівень, який складається з двох нейронів: перший відповідає за прискорення, другий за можливість автомобіля повертати вліво або вправо. Також у мережі немає чітко визначеної кількості прихованих рівнів та нейронів, які там розташовані. Всі ці дані у симуляції може визначати сам користувач. На вхід до нейронної мережі подаються значення отримані з лазерів. Лазери вимірюють відстань до перешкоди та подають значення на вхід до нейронної мережі. Обрана сигмоїдна функція активації. Для навчання нейронної мережі використовується алгоритм підсиленого навчання, а саме генетичний алгоритм, який застосовується до кожного автомобіля, починаючи зі створення власного списку генів кожного автомобіля. У мережі кожного автомобіля кількість генів дорівнює кількості ваг. Для першого покоління ваги задані випадковим чином. Оскільки це симулятор, була створена узагальнена нейронна мережа з великою кількістю налаштувань, з можливістю змінювати її структуру: можна змінити кількість нейронів вхідного рівня, які залежать від кількості лазерів у автомобіля, їх дальнобійність, висоту, на якій вони визначають перешкоди, поле їх видимості; змінювати кількість прихованих рівнів та кількість нейронів, які там будуть розташовані; керувати процесом мутації, який використовується у генетичному алгоритмі; визначити значення мутації та діапазон варіацій, у якому значення можуть змінюватись; вмикати та вимикати функцію самозбереження, міняти швидкість автомобіля, прискорення, встановлювати максимальну та мінімальну швидкість, редагувати параметри, які відповідають за обертання автомобіля та його плавність. Реалізація проекту надана у GitHub.
Ключові слова: симулятор, Unity 3D, нейронна мережа прямого поширення, функції активації, платформа GitHub.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Вступ і мета. Безпілотні, автономні, самокеровані так називають транспортні засоби, яким не потрібна людина в якості водія. Це не тільки легкові автомобілі, як здається на перший погляд, це ще й поїзди, вантажівки, сільськогосподарські машини тощо. За їх управління відповідає автоматизована система керування. В наступний час крупні автомобільні фірми перейшли від методів комп'ютерного моделювання на методи імітаційного моделювання на симуляторах, використовуя комп'ютерні моделі, щоб мати можливість використання отриманих даних у стендових іспитах [1, 2].

Самокерований автомобіль одна з найскладніших і, одночасно, найцікавіших тем в області штучного інтелекту сьогодні. Проблема є складною, оскільки система працює в частково спостережуваному, безперервному та динамічному середовищі. В такому середовищі система не тільки повинна піклуватися про завдання керування, але також повинна мати контекстну інформацію про навколишнє середовище в максимально можливій мірі.

Автономні транспортні засоби вимагають широкомасштабної розробки та тестування в широкому діапазоні сценаріїв, перш ніж вони зможуть бути розгорнуті та використані для реального світу. Однак тестові поїздки у реальному світі вимагають великих витрат часу та капіталу, і відтворити рідкісні та небезпечні сценарії практично неможливо.

Мета роботи – запропонувати підхід реалізації імітаційного моделювання на симуляторах руху транспортного засобу з навчанням, без використання комерційних програмних продуктів.

Симуляція. Вона дозволяє перевіряти поведінку автономних транспортних засобів у величезній кількості сценаріїв, середовищ, конфігурацій системи тощо. Це не робить випробувальні полігони застарілими, але може допомогти зосередитися саме на необхідних випробуваннях для перевірки результатів роботи розробників, тощо. Допомагає зберігати час та капітал, витрачений на випробування у реальному світі [6].

Автономний автомобіль здатен орієнтуватися у просторі, використовуючи різні прилади і методи, такі як радар, лідар, GPS, одометри, комп'ютерний зір та інші. Сучасні системи керування здатні інтерпретувати інформацію з сенсорів аби визначити правильний напрямок руху, а також ідентифікувати перешкоди й відповідні покажчики. Автономні автомобілі мають системи керування, які здатні аналізувати сенсорні дані і розрізняти об'єкти на дорозі, що є дуже важливим при слідуванні маршруту до бажаної точки призначення, бо автомобіль повинен уникати зіткнень з різними об'єктами.

Нині автомобільна індустрія зазнає істотної трансформації: найбільші виробники машин спільно з IT і телеком-розробниками йдуть до створення транспортних засобів з можливістю повністю автономного керування. Тренд вже очевидний в майбутньому безпілотний транспорт стане масовим явищем, але на шляху до епохи повністю автономних автомобілів ще належить вирішити масу завдань.

Автономні транспортні засоби вимагають широкомасштабної розробки та тестування в широкому діапазоні сценаріїв, перш ніж вони зможуть бути розгорнуті та використані для реального світу. Однак тестові поїздки у реальному світі вимагають великих витрат часу та капіталу, і відтворити рідкісні та небезпечні сценарії практично неможливо. Саме для цього і потрібні симулятори.

Наприклад, є симулятор NVIDIA DRIVE Sim [4]. Зі слів розробників, NVIDIA DRIVE Sim вирішує ці проблеми завдяки масштабованій, фізично точній та різноманітній симуляційній платформі. Завдяки DRIVE Sim розробники AV можуть підвищити продуктивність, ефективність та покриття тестів, просуваючи час виходу на ринок, мінімізуючи реальне водіння.

DRIVE Sim використовує високоточну та фізично точну імітацію для створення безпечного, масштабованого та економічно вигідного способу вивести на наші дороги автономні машини. Він використовує основні технології NVIDIA, включаючи NVIDIA RTX, Omniverse та AI, щоб забезпечити потужну хмарну обчислювальну платформу, здатну генерувати широкий спектр реальних сценаріїв розвитку та перевірки AV. DRIVE Sim може генерувати набори даних, щоб навчити систему сприйняття автомобіля або перевірити процес прийняття рішень. Він також може бути підключений до AV-стеку в конфігураціях програмного забезпечення в циклі або апаратного забезпечення в циклі для перевірки інтеграції системи. Його можно запускати як локально, так і з використанням хмарних технологій.

Однак проблема полягає у тому, що доступ до цього симулятора надається лише обмеженому колу розробників і він буде у ранньому доступі не раніше ніж улітку 2021го.

Є інші приклади симуляторів з відкритим кодом, наприклад CARLA [5]. CARLA була розроблена з нуля для підтримки розвитку, навчання та перевірки систем автономного водіння. На додаток до коду та протоколів з відкритим кодом, CARLA пропонує відкриті цифрові активи (міські плани, будівлі, транспортні засоби), які були створені для цієї мети та можуть вільно викори-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

стовуватися. Симуляційна платформа підтримує гнучку специфікацію сенсорних наборів, умов навколишнього середовища, повний контроль усіх статичних та динамічних факторів, створення карт тощо. Та навіть у таких версіях, з відкритим кодом, все ще залишаються проблеми з ліцензіями і розробнику доволі проблематично щось змінити, доробити, переписати або використати у власному продукті.

Відмітимо роботу [3], де на наш погляд добре надано огляд автономних транспортних засобів.

Інструмент Unity 3D. Для симулятору використано Unity 3D – багатоплатформовий інструмент для розробки багатовимірних додатків, симуляторів, ігор, тощо.

Ще одна особливість це комбіновані пакети елементів, що полегшує створення прототипів та не використовує вузли дерева успадкування. Це актуально для розробників ігрових додатків.

Основний недолік це використання схем у складних додатках. Багатокомпонентні сцени дуже ускладнюють візуальну частину роботи у редакторі. Також є проблема з зовнішніми бібліотеками, які розробник повинен підключати власноруч, що викликає труднощі та витрачає багато часу. Також багатоплатформенність це завжди компроміс у плані продуктивності, пам'яті, стабільності застосунку на усіх операційних системах та девайсах.

Однак незважаючи на все це, Unity 3D найкраще підходить для мови C# та середовища програмування Visual Studio, бо все це вже інтегроване в Unity, а багатоплатформенність дозволяє використовувати симулятор на різних девайсах та операційних системах.

Симулятор розроблений у програмі Unity 3D, для цього у Unity створена сцена для симуляції «рис. 1». Для створення сцени використовувались стандартні засоби редактору разом з використанням базових ассетів, наприклад, 3D модель автомобіля, та безкоштовні ассети інших користувачів, які можна завантажувати зі спеціального магазина Unity Asset Store, такі як замки, деякі гори, тощо. Для освітлення сцени були використані спеціальні компоненти освітлення, разом з нічним скайбоксом, який теж містить у собі компоненти освітлення.

Трек створений зі звичайних прямокутних об'єктів з параметрами, які роблять звичайний прямокутник схожим на стіну. До цих стін (перешкод) була додана функція колізії, яка обробляє зіткнення авто з перешкодами та знищує автомобіль.

Усі об’єкти та компоненти, які повинні безпосередньо приймати участь у симуляції, об’єднані головним класом контроллером. Він безпосередньо відповідає за створення нових поколінь у симуляції, на початку кожної ітерації. Містить у собі налаштування для завдання кількості агентів, які будуть брати участь у симуляції. Головний клас приймає у якості параметра об’єкт автомобіля, для якого він буде запускати симуляцію та містить скрипіт, написаний для обробки усіх сценаріїв, визначених у симуляції, запуску нових ітерації, ініціалізації камери, UI-компонентів, тощо.

Рисунок 1 - Сцена симуляції у Unity 3D

Нейронна мережа. У симуляції використовується нейронна мережа прямого поширення. Вона немає чіткої кількості нейронів вхідного рівня. Також у мережі немає чітко визначеної кількості прихованих рівнів та нейронів, які там розташовані. Всі ці дані у симуляції може визначати сам користувач. Мережа має тільки сталий вихідний рівень, який складається з двох нейронів перший відповідає за прискорення автомобіля, другий відповідає за можливість автомобіля повертати вліво або вправо «рис. 2».

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

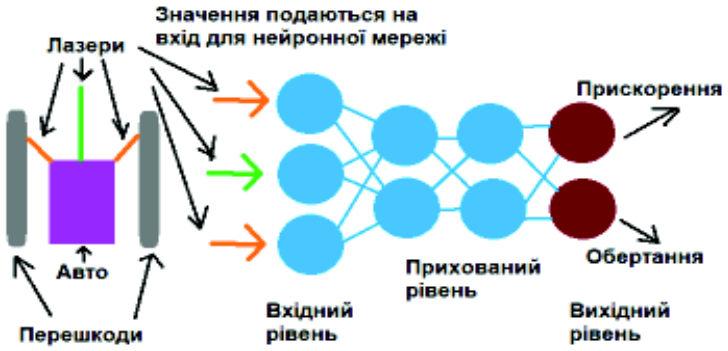

Рисунок 2 - Модель нейронної мережі для трьох лазерів

На вхід до нейронної мережі подаються значення отримані з лазерів. Лазери вимірюють відстань до перешкоди та подають значення на вхід до нейронної мережі. Лазери можуть приймати значення у діапазоні [0, 1]. До речі, саме тому була обрана сигмоїдна функція активації, оскільки вона існує саме у діапазоні [0, 1] та особливо добре підходить для цього випадку. Якщо у полі зору лазеру (це відстань яку можна регулювати у симуляторі) немає перешкоди він подає значення 1 на вхід нейронної мережі. Якщо ж перешкода є у його полі зору, то чим менша відстань до перешкоди, тим менше значення, яке повертає лазер та подає на вхід нейронної мережі. Далі після обчислень, нейрони вихідного рівня мають два значення, одне прискорення, друге обертання. Ці два значення використовуються у стандартній фізиці Unity та шляхом геометричних перетворень рухають авто вперед та повертають його. Такі обчислення робляться для кожного авто на кожне оновлення кадрового буферу.

Навчання нейронної мережі. Для навчання нейронної мережі використовується алгоритм підсиленого навчання, а саме генетичний алгоритм, який застосовується до кожного автомобіля (агента). Згідно з закону Дарвіна, сутність (у роботі автомобіль) з найкращими генами, виживаючи, поширює свої гени наступному поколінню, яке поширює їх далі і далі, постійно розвиваючись.

Застосовуючи цей принцип для автомобілів, починаємо зі створення власного списку генів для кожного автомобіля. У нейронній мережі кожного автомобіля кількість генів дорівнює кількості ваг. Для першого покоління ваги будуть задані випадковим чином. В плані програмування - ваги (гени) представлені масивом, який ініціалізується у діапазоні від [-1,1]. Значення для діапазону у симуляції можна змінювати. Далі обираються найкращі агенти, тобто автомобілі, які прожили найбільше часу у симуляції. У наступній ітерації (поколінні), коли мережа буде оновлюватись, значення ваг будуть залишатися

статичними, а значення нейронів, під час симуляції, будуть змінюватись, оскільки вони залежать від нейронів вхідного рівня та ваг мережі.

Генетичний алгоритм розділений на різні розділи:

•випадкова ініціалізація ДНК;

•вибір кращих автомобілів залежно від функції пристосування;

•перехрещення генів батьків;

• мутація нової ДНК.

Оскільки це симулятор, була створена узагальнена нейронна мережа, з великою кількістю налаштувань, з можливістю змінювати її структуру. Наприклад, можна змінити кількість нейронів вхідного рівня, які залежать від кількості лазерів у автомобіля. Якщо авто має 3 лазери, які вимірюють відстань, то буде 3 нейрони вхідного рівня, якщо ж авто має 10 лазерів, то буде 10 нейронів вхідного рівня. Також можна змінювати кількість прихованих рівнів та кількість нейронів, які там будуть розташовані. Кількість нейронів вихідного рівня залишається незмінною, бо це є прискоренням та обертанням для автомобіля. Тільки зі зміною логіки нейронної мережі можна буде змінити ці параметри.

Для автомобіля можна вмикати та вимикати функцію самозбереження. Міняти швидкість автомобіля, прискорення, встановлювати максимальну та мінімальну швидкість, редагувати параметри, які відповідають за обертання автомобіля та його плавність.

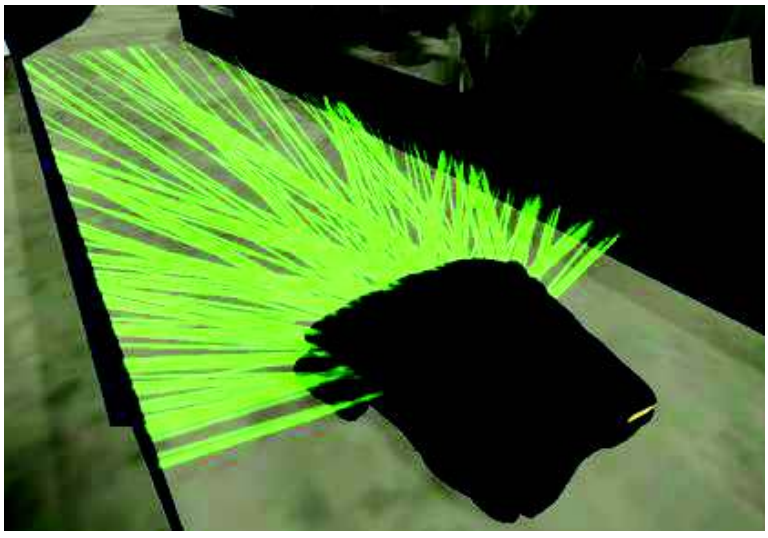

Рисунок 3 - Початок симуляції. П’ятдесят агентів

Як було сказано вище, можна змінювати кількість лазерів, їх дальнобійність, висоту, на якій вони визначають перешкоди, поле їх

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

видимості, наприклад, вони можуть бути налаштовані для пошуку перешкод тільки спереду або повністю з усіх боків.

Також можна керувати процесом мутації, який використовується у генетичному алгоритмі, можна визначити значення мутації та діапазон варіацій, у якому значення можуть змінюватись.

У симуляторі можна змінювати кількість агентів. Зі збільшенням кількості агентів, ми збільшуємо швидкість тренування, особливо на перших ітераціях, оскільки кожен автомобіль має свою власну мережу та власні значення для мережі та через це ми покриваємо більше значень, які як раз можуть бути тими, які нам потрібні та для їх знаходження знадобиться менше часу «Рис.3».

У роботі кожен автомобіль має власну нейронну мережу, і вона оновлюється кожного разу, як починається нове покоління. Цей алгоритм оновлення називається «feedforward», тобто алгоритм прямого поширення. Автомобілі будуть мати в якості вхідного рівня відстані до стін, що оточують машини За допомогою даних вхідного рівня та ваг, мережа розраховує вихідні данні, які будуть обертанням і прискоренням для автомобілів.

Висновки. Створений у роботі симулятор це базова, безкоштовна, початкова версія. Симулятор може бути завантаженим будь-яким розробником з платформи GitHub [7] та використаний для реалізації та тестування різних алгоритмів навчання для нейронної мережі, у тому числі й власноруч створених. Так розробник, який завантажив симулятор, зможе порівняти результати роботи програми, провести аналіз та виявити найкращі алгоритми для нейронної мережі прямого поширення. Також можна буде порівняти результати роботи програми та швидкість навчання нейронної мережі, змінивши параметри нейронної мережі, параметри лазерів, автомобілів, тощо. Розробник може вільно вносити зміни до коду та перевіряти потрібний йому функціонал. Базова версія симулятору, може бути взята за основу для створення більш складного та функціонального симулятора. Фрагменти коду цього симулятора або об’єкти сцени Unity, можуть бути використані в інших додатках. Після внесення змін, розробник може запросити ревью доданого ним функціоналу та коду та створити запит на додавання його коду у репозиторій. Інші розробники зможуть скачати вже більш функціональний симулятор та у майбутньому теж розширити його функціонал.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Mercedes-Benz Innovation Vehicle Developing. [Електроний ресурс/Electronic resource]/–Режим доступу/Access mode: 22.07.2018. https://www.mercedes-benz.com/en/mercedes-benz/next/advancedengineering/

2.Emanuele Obialero. A Refined Vehicle Dynamics Model for Driving Simula…rs // Charhalmers University of Technology/Göteborg, Sweden 2013. Master's thesis, P. 120.
3.Shaoshan Liu, Liyun Li, Jie Tang, Shuang Wu, JeanLuc Gaudiot. «Creating Autonomous Vehicle Systems». Morgan & Claypool Publishers, 2018. - P. 191
4. Закритий симулятор NVIDIA DRIVE Sim. [Електроний ресурс/Electronic resource]/ https://developer.nvidia.com/drive-sim-early-access-program.
5. Симулятор з відкритим кодом Carla. [Електроний ресурс/Electronic resource]/ CARLA Simulator.
6. Михайлов В.Г. О некоторых подходах моделирования автомобиля на симуляторах // Системный анализ и прикладная информатика. №3, -2019. - с. 29-35.
7. [Electronic resource] https://github.com/Evyshkav/AVSimulator

REFERENCES

1. Mercedes-Benz Innovation Vehicle Developing. https://www.mercedes-benz.com/en/mercedes-benz/next/advancedengineering/ [Electronic resource]– Access mode: 22.07.2018.
2.Emanuele Obialero. A Refined Vehicle Dynamics Model for Driving Simula…rs // Charhalmers University of Technology/Göteborg, Sweden 2013. Master's thesis, P. 120.
3.Shaoshan Liu, Liyun Li, Jie Tang, Shuang Wu, JeanLuc Gaudiot. «Creating Autonomous Vehicle Systems». Morgan & Claypool Publishers, 2018. - P. 191
4. Closed Simulator NVIDIA DRIVE Sim. https://developer.nvidia.com/drive-sim-early-access-program.
5. Open Source Simulator Carla. CARLA Simulator.
6. Muhaylov B.G. O nekotopukh podkhodakh modelirovaniya avtomobilya na ci-mylyatorakh // Sistemnui analiz i prikladnaya informatika. №3, -2019. - c. 29-35.
7. https://github.com/Evyshkav/AVSimulator

Received 06.10.2021.
Accepted 14.10.2021.

On one approach to the development of a simulator of the movement of an autonomous vehicle with training

The article deals with the use of simulators for controlling the movement of an autonomous vehicle and development of a new simulator. The approach to creating a simulator of motion of a vehicle in the C# programming language is described. In the development for the implementation of simulation scenes used Unity 3D multi-platform tool is used in the development. The simulation uses direct propagation neural network that does not have a clear number of input level neurons, having only a constant output level, consisting of two neurons: the first one is responsible for acceleration, the second one is responsible for the ability of the car to turn left or to the right. Also, there is no clearly defined number of hidden levels and neurons located there. All this data in the simulation can be determined by the user. The input to the neural network values received from lasers. The lasers measure the distances to obstacles

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

and feed the values to the input of the neural network. A sigmoidal activation function is implemented. To train the neural network an augmented learning algorithm is used, namely, a genetic algorithm applied to each vehicle, starting with the creation of each vehicle's own list of genes. In the network of each vehicle the number of genes is equal to the number of weights. For the first generation, the weights are set randomly. For the simulation, a generalized neural network with a large number of settings, with it is possible to change its structure: it is possible to change the number input level neurons that depend on the number of lasers at vehicle, their range, the height at which they detect interference, field of their visibility; you can change the number of hidden levels and the number of neurons that will be located there; control the mutation process used in the genetic algorithm; define the value of the mutation and the range of variation at which values can be varied; turn on and off the self-preservation, change vehicle speed, acceleration, set the maximum and minimum speed, edit the parameters responsible for the rotation of the car and its smoothness. Implementation of the project is provided on GitHub. The simulator can be downloaded by any developer from GitHub and can be used to implement and test various neural network training algorithms, including work of your own design.

Об одном подходе для разработки симулятора движения автономного транспортного средства с обучением

В данной работе рассматривается проблема использования симуляторов для управления движением автономным транспортным средством и разработки нового симулятора, который использует язык программирования C#, среду реализации сцен симуляции в Unity 3D. Моделирование использует нейронную сеть с прямым распространением. Созданный симулятор является базовой, бесплатной и начальной версией. Симулятор может скачиваться любым разработчиком с GitHub и использоваться для реализации и тестирования различных алгоритмов обучения нейронной сети, в том числе работы собственной разработки.

Зайцев Вадим Григорович – кандидат фізико-математичних наук, доцент, доцент кафедри комп'ютерних технологій Дніпровського національного університету імені Олеся Гончара.
Булатецкий Євген Іванович – студент Дніпровського національного університету імені Олеся Гончара.

Зайцев Вадим Григорьевич – кандидат физико-математичесних наук, доцент, доцент кафедри компьютерних технологій Днепровського национального университета имени Олеся Гончара.
Булатецкий Евгений Иванович – студент Днепровського национального университета имени Олеся Гончара.

Zaytsev Vadym Grigorovich - Ph.D., Associate Professor, Department of computer Technologies, Oles Honchar Dnipro National University.
Bulatetskyi Yevhenii - student, Oles Honchar Dnipro National University.