

การพัฒนาระบบจอตลอดอัตโนมัติโดยใช้การเรียนรู้แบบเสริมกำลังเชิงลึกจากสภาพแวดล้อมจำลอง

วรพงษ์ ศรีสอาด

TNI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

บัณฑิตศึกษา สาขาเทคโนโลยีวิศวกรรม

สถาบันเทคโนโลยีไทย-ญี่ปุ่น

ปีการศึกษา 2568

DEVELOPMENT OF AN AUTONOMOUS PARKING SYSTEM USING DEEP REINFORCEMENT
LEARNING IN A SIMULATED ENVIRONMENT

Vorrapong Srisaard

TNI

A Thesis Submitted in Partial Fulfillment of Requirements
for The Degree of Master of Engineering Program in Engineering Technology
Graduate Studies

Thai-Nichi Institute of Technology

Academic Year 2025

หัวข้อวิทยานิพนธ์

การพัฒนาระบบจอตลอดอัตโนมัติโดยใช้การเรียนรู้แบบเสริม

กำลังเชิงลึกจากสภาพแวดล้อมจำลอง

โดย

วรพงษ์ ศรีสอาด

สาขาวิชา

เทคโนโลยีวิศวกรรม

อาจารย์ที่ปรึกษาวิทยานิพนธ์

ดร.ศิวพงษ์ นิลวงษ์

บัณฑิตศึกษา สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้บัณฑิตวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่ง
ของการศึกษาตามหลักสูตรปริญญาโทบริหารธุรกิจ

รองอธิการบดีฝ่ายวิชาการ

(รองศาสตราจารย์ ดร.วรสาร ศรีเชวงทรัพย์)

วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

ประธานกรรมการ

(ดร.ณัฐกิตต์ จิตรเอื้อกุล)

กรรมการ

(ดร.ชัชชัย วรรณบูรณ์)

กรรมการ

(ดร.อัฒนา เชนโต๊ะ)

อาจารย์ที่ปรึกษาวิทยานิพนธ์

(ดร.ศิวพงษ์ นิลวงษ์)

วรพงษ์ ศรีสอาด : การพัฒนาระบบจอดรถอัตโนมัติโดยใช้การเรียนรู้แบบเสริมกำลังเชิงลึก
จากสภาพแวดล้อมจำลอง. อาจารย์ที่ปรึกษา : ดร.ศิวพงษ์ นิลวงษ์, 85 หน้า.

การจอดรถเป็นหนึ่งในกิจกรรมที่ผู้ใช้งานพาหนะทุกคนต้องประสบพบเจอและมีความท้าทาย โดยเฉพาะอย่างยิ่งในปัจจุบันที่จำนวนผู้ใช้รถยนต์มีการเพิ่มขึ้นอย่างต่อเนื่อง ส่งผลให้การค้นหาที่จอดรถเป็นเรื่องที่ยากลำบากขึ้น และยังเพิ่มความเสี่ยงต่ออันตรายจากการชนหรือเฉี่ยวชนขณะเข้าจอด เพื่อเป็นการยกระดับความสะดวกและรวดเร็วในการจอดรถ เทคโนโลยีระบบปัญญาประดิษฐ์ (AI) สำหรับการจอดรถอัตโนมัติจึงเข้ามามีบทบาทสำคัญ การวิจัยนี้มุ่งเน้นการนำเทคโนโลยี AI มาใช้ในการพัฒนาและฝึกอบรมโมเดลสำหรับระบบจอดรถอัตโนมัติ โดยใช้สภาพแวดล้อมจำลองที่สร้างขึ้นจากแพลตฟอร์ม Unity ซึ่งเป็นเครื่องมือที่มีประสิทธิภาพในการจำลองสถานการณ์เสมือนจริง โมเดล AI จะได้รับการฝึกฝนและพัฒนาให้เรียนรู้พฤติกรรมจอดรถโดยใช้เทคนิค Proximal Policy Optimization (PPO) ซึ่งเป็นหนึ่งในรูปแบบของ Reinforcement Learning (RL) โดยงานวิจัยในครั้งนี้ได้ทำการทดลองบนสภาพแวดล้อมจำลอง เทคนิคนี้เลียนแบบวิธีการเรียนรู้ของมนุษย์ โดยเน้นการเรียนรู้ผ่านการลองผิดลองถูกและการได้รับผลตอบแทนจากการกระทำนั้นๆ คล้ายกับการฝึกฝนสิ่งมีชีวิตด้วยการให้รางวัลเมื่อกระทำได้อย่างถูกต้อง และลดรางวัลหรือลงโทษเมื่อกระทำผิดพลาด เป้าหมายหลักของการฝึกฝนโมเดล PPO คือการลดระยะห่างระหว่างรถกับสิ่งกีดขวาง และเพิ่มความแม่นยำในการเข้าช่องจอด เพื่อประเมินประสิทธิภาพในการจอดรถในสถานการณ์จำลองที่หลากหลาย เช่น การจอดรถระหว่างรถสองคัน และการจอดในช่องจอดที่แคบ ผลการวิจัยที่คาดหวังจะนำไปสู่การพัฒนาระบบช่วยเหลือการจอดรถอัตโนมัติที่สามารถทำงานได้อย่างมีประสิทธิภาพและปลอดภัยในสภาพแวดล้อมจริง ซึ่งจะช่วยเพิ่มความสะดวกสบายและลดความเสี่ยงให้กับผู้ขับขี่ในอนาคต

บัณฑิตศึกษา

สาขาวิชา เทคโนโลยีวิศวกรรม

ปีการศึกษา 2568

ลายมือชื่อนักศึกษา.....

ลายมือชื่ออาจารย์ที่ปรึกษา.....

VORRAPONG SRISAARD: DEVELOPMENT OF AN AUTONOMOUS PARKING SYSTEM
USING DEEP REINFORCEMENT LEARNING IN A SIMULATED ENVIRONMENT. ADVISOR:
DR. SIVAPONG NILWONG, 85 PP.

Parking is one of the most challenging activities for all vehicle users, especially with the ever-increasing number of cars. Finding a parking space becomes more difficult and increases the risk of collisions while parking. To improve the convenience and speed of parking, Artificial Intelligence (AI) for automated parking plays a crucial role. This research focuses on developing and training a model for an automated parking system using AI technology. A simulated environment built on the Unity platform, a powerful tool for simulating realistic scenarios, will be used. The AI model will be trained to learn parking behavior using Proximal Policy Optimization (PPO), a form of Reinforcement Learning (RL). This technique mimics human learning by focusing on trial and error and rewarding actions, similar to training living organisms by rewarding correct actions and decreasing rewards or penalties for errors. The primary goal of PPO training is to reduce the distance between vehicles and obstacles and increase parking accuracy. The performance of the AI model will be evaluated in various simulated scenarios, such as parking between two cars and parking in narrow spaces. The expected research results will lead to the development of an efficient and safe automated parking assistance system in real-world environments, thereby increasing convenience and reducing risks for drivers in the future.

Graduate Studies

Field of Engineering of Technology

Academic Year 2025

Student's Signature.....

Advisor's Signature.....

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จลุล่วงได้ด้วยดีเนื่องจากได้รับความกรุณาอย่างสูงจากอาจารย์ที่ปรึกษา ดร.ศิวพงษ์ นิลวงษ์ ที่กรุณาให้คำปรึกษาและคำแนะนำเกี่ยวกับงานวิจัย และแนะนำแนวทางในการ แก้ไขปัญหาต่างๆ ด้วยความเอาใจใส่เป็นอย่างยิ่ง รวมถึงอาจารย์ทุกท่านที่มีได้กล่าวนามที่ให้ความ อนุเคราะห์ให้ความช่วยเหลือในด้านต่างๆ และคอยให้คำแนะนำที่เป็นประโยชน์กับงานวิจัย ทำให้ งานวิจัยนี้สำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ผู้วิจัยหวังเป็นอย่างยิ่งว่า งานวิจัยฉบับนี้จะเป็นประโยชน์ต่อผู้ศึกษาไม่มากนักน้อย ซึ่งขอกราบขอบพระคุณ ครอบครัว พี่ๆ ที่ทำงาน และเพื่อนๆ ทุกท่านทั้งผู้ที่กล่าวนามและมีได้กล่าวนาม ที่ คอยให้กำลังใจ ให้คำปรึกษาและสนับสนุนในการทำงาน จนดำเนินงานสำเร็จลุล่วงไปได้ด้วยดี

วรพงษ์ ศรีสอาด

TNI

สถาบันเทคโนโลยีไทย-ญี่ปุ่น
TAIJI - NICHU INSTITUTE OF TECHNOLOGY

สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	ง
บทคัดย่อภาษาอังกฤษ.....	จ
กิตติกรรมประกาศ.....	ฉ
สารบัญ.....	ช
สารบัญตาราง.....	ฌ
สารบัญรูป.....	ญ
บทที่	
1 บทนำ.....	1
1.1 ที่มาและความสำคัญ.....	1
1.2 ความสำคัญของการวิจัย.....	2
1.3 วัตถุประสงค์ของการวิจัย.....	2
1.4 ขอบเขตของการวิจัย.....	2
1.5 ผลที่คาดว่าจะได้รับ.....	3
2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง.....	4
2.1 ทฤษฎีที่เกี่ยวข้อง.....	4
2.2 ทบทวนวรรณกรรม.....	15
3 วิธีการวิจัย.....	25
3.1 การศึกษาอัลกอริทึมและทฤษฎีที่เกี่ยวข้อง.....	25
3.2 การออกแบบและสร้างสภาพแวดล้อมจำลอง.....	27
3.3 การทดสอบและวิเคราะห์ผล.....	30
4 ผลการศึกษาและการอภิปรายผล.....	31
4.1 ภาพรวมของระบบจอตลอดอัตโนมัติ.....	31
4.2 โมเดลอัลกอริทึมและการออกแบบฟังก์ชันรางวัล.....	31
4.3 การวิเคราะห์ความเสถียรของกระบวนการเรียนรู้.....	37

สารบัญ (ต่อ)

บทที่		หน้า
4	4.4 การทดลอง.....	40
	4.5 การวิเคราะห์ผลการทดลอง.....	43
5	สรุปผลและข้อเสนอแนะ.....	45
	5.1 สรุปผล.....	45
	5.2 ปัญหาและข้อจำกัดของการวิจัย.....	45
	5.3 ข้อเสนอแนะ.....	45
	บรรณานุกรม.....	47
	ภาคผนวก.....	52
	ภาคผนวก ก. ชุดคำสั่งภาษา C#.....	53
	ภาคผนวก ข. บทความการประชุมวิชาการ.....	78
	ประวัติย่อผู้วิจัย.....	85

สารบัญตาราง

ตาราง	หน้า
2.1 สรุปลงานวิจัยที่เกี่ยวข้องกับแนวทางที่นำเสนอ	22
3.1 การตั้งค่าคุณสมบัติของสภาพแวดล้อมการจำลอง	29
3.2 การตั้งค่ารถยนต์จำลอง	29
4.1 การตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO สำหรับจุดทางตรง	34
4.2 การตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO สำหรับจุด 45 องศา	34
4.3 การกำหนดค่ารางวัลพื้นฐานในระบบจำลอง	35
4.4 การกำหนดรางวัลแบบจูงใจ	36
4.5 ผลการเปรียบเทียบประสิทธิภาพระบบจอดรถอัตโนมัติในสถานการณ์จอด เข้าช่อง 45 องศา	41

TNI

TAI - NICH I INSTITUTE OF TECHNOLOGY

สารบัญรูป

รูป	หน้า
2.1 ตัวอย่างสถาปัตยกรรมของการเรียนรู้เชิงลึก.....	5
2.2 ตัวอย่างสถาปัตยกรรม Deep reinforcement learning	7
2.3 ตัวอย่างของ MDP อย่างง่ายที่มีสามสถานะ (วงกลมสีเขียว) และสองการกระทำ (วงกลมสีส้ม) พร้อมสองรางวัล (ลูกศรสีส้ม).....	8
2.4 ตัวอย่างสถาปัตยกรรม Reinforcement Learning.....	9
2.5 ตัวอย่างสถาปัตยกรรมของ Deep Q Network	10
2.6 Convolutional Layer และ การทำ Flattening	12
2.7 ตัวอย่างสถาปัตยกรรม Proximal Policy Optimization	14
2.8 โปรแกรม Unity สำหรับสร้างสภาพแวดล้อมจำลอง	15
3.1 Convolutional Layer และ การทำ Flattening	27
3.2 ภาพจำลองสภาพแวดล้อมแบบมีจุดเริ่มต้น 1 จุดในมุมมองด้านบน	28
3.3 ภาพจำลองสภาพแวดล้อมแบบมีจุดเริ่มต้น 3 จุดในมุมมองด้านบน	28
4.1 แผนผังการทำงานของระบบจอตลอดอัตโนมัติ	31
4.2 แผนผังโมเดล PPO ของตัวแทนและการไหลของข้อมูล.....	32
4.3 ผลการทดลองการจอตลอดทางตรงโดยใช้ระบบรางวัลแบบให้ครั้งเดียว	37
4.4 ผลการทดลองการจอตลอดทางตรงโดยใช้ระบบรางวัลแบบมีแรงจูงใจ	38
4.5 ผลการทดลองการจอตลอดโดยสุ่มจุดเกิด 3 จุดโดยใช้ระบบรางวัลแบบให้ครั้งเดียว	39
4.6 ผลการทดลองการจอตลอดโดยสุ่มจุดเกิด 3 จุดโดยใช้ระบบรางวัลแบบมีแรงจูงใจ	40
4.7 ตัวอย่างการเข้าจอตที่สำเร็จทุกเงื่อนไข.....	41
4.8 ตัวอย่างการเข้าจอตที่สำเร็จแต่ไม่ผ่านเงื่อนไขการจอตให้ตรงกับเป้าหมาย.....	42
4.9 ตัวอย่างการเข้าจอตที่ไม่สำเร็จโดยชนกับแพง.....	42
4.10 ตัวอย่างการเข้าจอตที่ไม่สำเร็จโดยชนกับรถยนต์ที่จอตอยู่	43
4.11 รูปของ Agent ในมุมมองบุคคลที่ 1 (a) Agent เมื่อเริ่ม Episode (b) Agent เคลื่อนที่ เข้าใกล้เป้าหมาย (c) Agent เมื่อถึงเป้าหมาย.....	44

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันการขยายตัวของเขตเมืองและจำนวนยานพาหนะที่เพิ่มขึ้นอย่างรวดเร็วส่งผลให้ปัญหาการจราจรทวีความรุนแรงขึ้นอย่างต่อเนื่อง โดยเฉพาะอย่างยิ่งปัญหาการจราจรติดขัดที่เกิดจากการวนหาที่จอดรถ ซึ่งจากการศึกษาและรวบรวมข้อมูลพบว่าในพื้นที่ย่านธุรกิจที่มีการจราจรหนาแน่น ยานพาหนะกว่าร้อยละ 30 บนท้องถนนคือยานพาหนะที่กำลังวนหาที่จอดรถ [1] ปัญหานี้ไม่เพียงแต่ก่อให้เกิดการสูญเสียเวลาของผู้ขับขี่ แต่ยังส่งผลกระทบต่อตรงต่อการสิ้นเปลืองเชื้อเพลิงและการปล่อยมลพิษทางอากาศ โดยเฉพาะก๊าซคาร์บอนไดออกไซด์ (CO₂) และฝุ่นละอองขนาดเล็ก ซึ่งส่งผลกระทบต่อสภาพแวดล้อมและสุขภาพของประชาชนในระยะยาว [2] เพื่อแก้ไขปัญหาดังกล่าว เทคโนโลยีรถยนต์ไร้คนขับและระบบช่วยจอดอัตโนมัติ จึงถูกพัฒนาขึ้นเพื่อเพิ่มประสิทธิภาพในการบริหารจัดการพื้นที่จอดรถและลดความผิดพลาดจากมนุษย์ อย่างไรก็ตาม การพัฒนาระบบจอดอัตโนมัติในสถานการณ์จริงนั้นมีความซับซ้อนสูง เนื่องจากรถยนต์ต้องทำงานในสภาพแวดล้อมที่มีพลวัตและไม่สามารถคาดเดาได้ เช่น การเคลื่อนที่ของรถคันอื่น หรือสิ่งกีดขวางต่างๆ อีกทั้งการใช้วิธีการควบคุมแบบดั้งเดิม ไม่ยืดหยุ่นเพียงพอต่อสถานการณ์ที่หลากหลาย [3] ในช่วงไม่กี่ปีที่ผ่านมา เทคโนโลยีปัญญาประดิษฐ์ (AI) โดยเฉพาะการเรียนรู้แบบเสริมกำลัง (Reinforcement Learning: RL) ได้รับความสนใจอย่างมากในการนำมาประยุกต์ใช้เพื่อแก้ปัญหการควบคุมที่ซับซ้อน โดยจากการศึกษาเปรียบเทียบพบว่า อัลกอริทึม Proximal Policy Optimization (PPO) มีประสิทธิภาพสูงและมีเสถียรภาพในการเรียนรู้ที่โดดเด่นกว่า เมื่อเทียบกับอัลกอริทึมแบบดั้งเดิมอย่าง Deep Q-Network (DQN) โดยเฉพาะในงานที่ต้องควบคุมพื้นที่การกระทำแบบต่อเนื่อง เช่น การควบคุมองศาพวงมาลัยและการเร่งความเร็ว ซึ่ง PPO สามารถจัดการกับความผันผวนของการอัปเดตค่าพารามิเตอร์ได้ดีกว่า ทำให้การเคลื่อนที่ของยานพาหนะมีความราบรื่นและแม่นยำยิ่งขึ้น [4], [5] นอกจากนี้ การเลือกใช้ข้อมูลภาพ จากกล้องเพียงอย่างเดียวในการประมวลผล ยังเป็นแนวทางที่ช่วยลดต้นทุนของระบบฮาร์ดแวร์ได้อย่างมีนัยสำคัญ เมื่อเทียบกับการใช้เซนเซอร์วัดระยะทางแบบเลเซอร์ (LiDAR) ที่มีราคาสูง [6] การใช้ข้อมูลภาพยังช่วยให้ระบบสามารถเรียนรู้บริบทของสิ่งแวดล้อมได้ใกล้เคียงกับการรับรู้ของมนุษย์ งานวิจัยนี้จึงมุ่งเน้นการพัฒนาแบบจำลองการจอดอัตโนมัติโดยประยุกต์ใช้อัลกอริทึม PPO ร่วมกับข้อมูลภาพภายในสภาพแวดล้อมจำลอง เพื่อสร้างตัวแทนปัญญาประดิษฐ์ (Agent) ที่สามารถเรียนรู้และตัดสินใจในการจอดรถได้อย่างแม่นยำและปลอดภัย ซึ่งจะเป็แนวทางสำคัญในการยกระดับเทคโนโลยียานยนต์ไร้คนขับในอนาคต

1.2 ความสำคัญของการวิจัย

การวิจัยนี้มีความสำคัญในการนำเสนอแนวทางการแก้ปัญหาการจราจรด้วยเทคโนโลยีปัญญาประดิษฐ์สมัยใหม่ ซึ่งมีความสำคัญดังนี้:

1.2.1 เพื่อเสนอแนวทางลดปัญหาการจราจรที่เกิดจากการวนหาที่จอดรถ ซึ่งจะช่วยลดการสิ้นเปลืองพลังงานเชื้อเพลิงและการปล่อยมลพิษสู่สิ่งแวดล้อม [2]

1.2.2 เพื่อแสดงศักยภาพของอัลกอริทึม PPO ในการควบคุมรถยนต์อัตโนมัติด้วยข้อมูลภาพ ซึ่งมีความซับซ้อนน้อยกว่าและต้นทุนต่ำกว่าระบบที่พึ่งพาเซนเซอร์ราคาแพง [6]

1.2.3 เพื่อเป็นต้นแบบในการพัฒนาระบบจราจรอัตโนมัติที่สามารถเรียนรู้และปรับตัวได้ด้วยตนเอง ผ่านกระบวนการลองผิดลองถูกในโลกจำลอง ซึ่งช่วยลดความเสี่ยงและค่าใช้จ่ายในการทดสอบกับรถยนต์จริง [7]

1.3 วัตถุประสงค์ของการวิจัย

1.3.1 เพื่อศึกษาและพัฒนาระบบจราจรอัตโนมัติสำหรับรถยนต์ขับเคลื่อนด้วยตัวเอง โดยใช้ข้อมูลภาพจากกล้องร่วมกับอัลกอริทึม PPO ภายในสภาพแวดล้อมจำลอง

1.3.2 เพื่อเปรียบเทียบประสิทธิภาพการเรียนรู้และพฤติกรรมของปัญญาประดิษฐ์ จากการกำหนดเงื่อนไขการให้รางวัล ในรูปแบบที่แตกต่างกัน 2 รูปแบบ

1.3.3 เพื่อประเมินความสามารถในการเข้าจอดในช่องจอดอย่างถูกต้องและปลอดภัยของระบบที่พัฒนาขึ้น

1.4 ขอบเขตของการวิจัย

เพื่อให้การศึกษามีเป็นไปตามวัตถุประสงค์ที่ตั้งไว้ ผู้วิจัยได้กำหนดขอบเขตของการดำเนินงานดังนี้:

1.4.1 สภาพแวดล้อมจำลอง: ดำเนินการสร้างและทดสอบบน Unity Game Engine ซึ่งเป็นเครื่องมือที่มีความสมจริงทางฟิสิกส์และรองรับการฝึกฝน ML-Agents ได้อย่างมีประสิทธิภาพ ช่วยลดความเสี่ยงจากการทดสอบจริง [8]

1.4.2 ข้อมูลนำเข้า: ใช้ข้อมูลภาพ จากกล้องที่ติดตั้งบนตัวรถจำลองเป็นเซนเซอร์หลักเพียงอย่างเดียว โดยไม่ใช้ข้อมูลจากเซนเซอร์วัดระยะทาง หรือ LiDAR

1.4.3 อัลกอริทึมที่ใช้: ใช้อัลกอริทึม PPO ในการฝึกฝน Agent เนื่องจากความเหมาะสมกับงานที่มีความต่อเนื่องของการกระทำ

1.4.4 การทดลอง: พัฒนาและทดสอบตัวแทนโดยแบ่งเงื่อนไขการให้รางวัลออกเป็น 2 รูปแบบ เพื่อเปรียบเทียบระยะเวลาและความแม่นยำในการเรียนรู้การจราจร

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 ได้ระบบจอตรอัตโนมัติที่ตัวแทนสามารถเรียนรู้และนำรถเข้าจอดได้อย่างถูกต้องผ่านกระบวนการลองผิดลองถูก

1.5.2 ได้ผลลัพธ์การทดลองที่แสดงถึงประสิทธิภาพของตัวแทนทั้ง 2 รูปแบบ ในการเข้าถึงช่องจอดตามเงื่อนไขที่กำหนด



บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การนำเทคโนโลยี AI และ ML ในหลาย ๆ เทคโนโลยีได้แสดงให้เห็นถึง การพัฒนาการที่โดดเด่น และความซับซ้อนในการคิดและวิเคราะห์ที่ใกล้เคียงกับมนุษย์ ซึ่งได้มีการนำเทคโนโลยีเหล่านี้ไปประยุกต์อย่างกว้างขวาง ทั้งในด้านการศึกษา สาธารณสุขและในด้านของยานยนต์ ซึ่งในกระบวนการปรับแต่งฝึกสอน AI และ ML เหล่านี้ในเกมส์ก็ช่วยให้เราสามารถสภาพแวดล้อมได้ตามที่ ต้องการและมีความปลอดภัย

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การเรียนรู้เชิงลึก

คือ รูปแบบหนึ่งของปัญญาประดิษฐ์ที่ใช้โครงข่ายประสาทเทียมเชิงลึกในการเรียนรู้รูปแบบ และโครงสร้างที่ซับซ้อนจากข้อมูลจำนวนมาก โดยอาศัยการปรับค่าพารามิเตอร์ภายในเครือข่ายผ่านกระบวนการที่เรียกว่า Backpropagation และ Gradient Descent ซึ่งในทางคณิตศาสตร์ การทำงานของแต่ละโหนดประสาท (Neuron) สามารถอธิบายได้ด้วยสมการ:

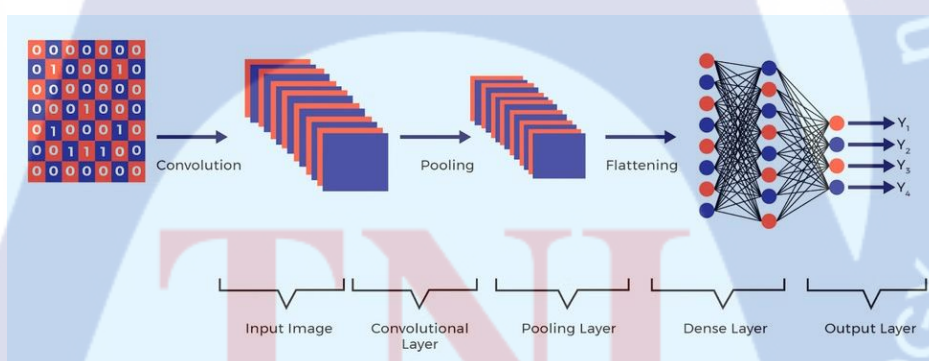
$$y = \sigma \sum_{i=1}^n w_i x_i + b \quad (1)$$

โดยที่ x_i คือข้อมูลอินพุต, w_i คือค่าน้ำหนัก (Weight), b คือค่าไบอัส (Bias), σ คือ Activation Function เช่น ReLU [9] หรือ Sigmoid [10] ที่ช่วยให้โมเดลสามารถเรียนรู้ความสัมพันธ์แบบไม่เป็นเชิงเส้นได้ การปรับปรุงค่าพารามิเตอร์จะใช้สมการการเคลื่อนที่ตามความชัน (Gradient Descent Update Rule) ดังนี้:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) \quad (2)$$

ซึ่ง θ_t คือ ค่าพารามิเตอร์ของโมเดลในรอบการอัปเดตปัจจุบัน, θ_{t+1} คือ ค่าพารามิเตอร์ของโมเดลในรอบการอัปเดตถัดไป, α คืออัตราการเรียนรู้และ $\nabla J(\theta_t)$ คือความชันของฟังก์ชันความสูญเสีย วิธีการนี้ช่วยให้โมเดลสามารถเรียนรู้ฟีเจอร์ต่างๆ ได้โดยอัตโนมัติจาก

ข้อมูลอินพุต แทนที่การออกแบบพีเจอร์เหล่านั้นด้วยมือ โครงสร้างหลักของ Deep Learning ประกอบด้วยหลายเลเยอร์ของโหนดประสาทเทียมที่ทำหน้าที่แปลงข้อมูลจากเลเยอร์หนึ่งไปยังอีกเลเยอร์หนึ่ง ซึ่งรวมถึง Convolutional Layer ที่ใช้สำหรับประมวลผลภาพ, Recurrent Layer ที่ใช้สำหรับวิเคราะห์ข้อมูลลำดับเวลา เช่น ข้อความและเสียง, และ Fully Connected Layer ที่ช่วยในการตัดสินใจขั้นสุดท้าย โมเดลยอดนิยมใน Deep Learning เช่น CNN, Recurrent Neural Networks (RNN) และ Transformer ได้ถูกนำไปใช้ในงานต่างๆ เช่น การรู้จำภาพ การแปลภาษา และการขับอัตโนมัติ โดยการรวม Deep Learning เข้ากับการเรียนรู้เสริมแรง ทำให้เกิดเทคนิค Deep Reinforcement Learning ซึ่งช่วยให้ AI สามารถเรียนรู้พฤติกรรมผ่านการโต้ตอบกับสภาพแวดล้อมและรับรางวัลจากการตัดสินใจที่ถูกต้อง Deep Learning จึงถือเป็นรากฐานสำคัญของเทคโนโลยี AI สมัยใหม่ที่ช่วยให้คอมพิวเตอร์สามารถเรียนรู้และตัดสินใจในระดับที่ซับซ้อนมากขึ้น ด้วยลักษณะการทำงานดังกล่าว การเรียนรู้เชิงลึกจึงมีความเหมาะสมอย่างยิ่งสำหรับการฝึกฝน AI ที่ต้องจัดการกับข้อมูลปริมาณมหาศาลและมีความซับซ้อนสูง โดยที่โมเดลสามารถสกัดคุณลักษณะได้เองโดยอัตโนมัติ ซึ่งให้ประสิทธิภาพที่โดดเด่นในงานประมวลผลภาพและการมองเห็นของคอมพิวเตอร์ งานวิเคราะห์ข้อมูลที่มีลำดับเวลา ตลอดจนงานที่ต้องการการจดจำรูปแบบและการตัดสินใจขั้นสุดท้ายที่มีความซับซ้อน



รูปที่ 2.1 ตัวอย่างสถาปัตยกรรมของการเรียนรู้เชิงลึก

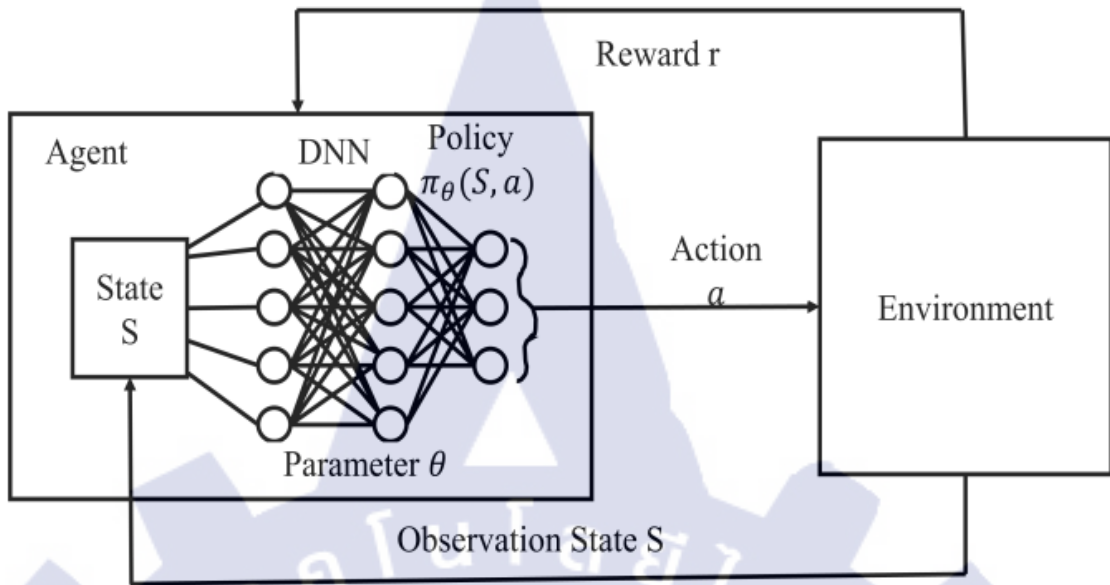
2.1.2 การเรียนรู้แบบเสริมเชิงลึก

คือ เทคนิคปัญญาประดิษฐ์ที่ผสมผสานระหว่างการเรียนรู้แบบเสริมแรง (Reinforcement Learning) และโครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks) เพื่อช่วยให้ Agent สามารถเรียนรู้และตัดสินใจในสภาพแวดล้อมที่ซับซ้อนที่มีมิติสูง โดยไม่ต้องมีแบบจำลองของระบบล่วงหน้า ในบริบทของการนำทางยานยนต์ไร้คนขับ DRL ถูกนำมาใช้เพื่อให้รถสามารถตัดสินใจได้อย่างอัตโนมัติ

ผ่านกระบวนการเรียนรู้จากประสบการณ์ โดยใช้อัลกอริทึมสำคัญ เช่น DQN ซึ่งใช้โครงข่ายประสาทเทียมในการประมาณค่าฟังก์ชัน $Q(s, a; \theta)$ แทนการใช้ตาราง Q-Table แบบดั้งเดิม โดยมีเป้าหมายเพื่อลดความผิดพลาดระหว่างค่า Q ที่ทำนายได้กับค่าเป้าหมาย ผ่านฟังก์ชันความสูญเสียดังสมการ:

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \quad (3)$$

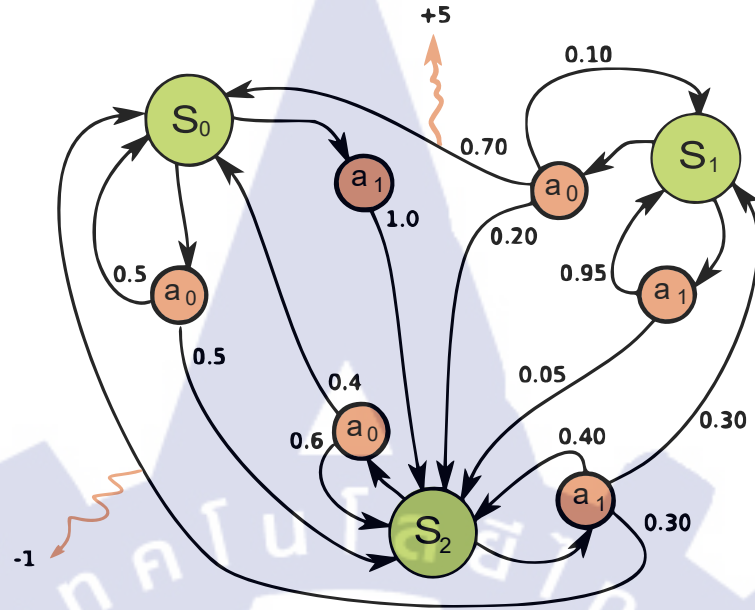
โดยที่ $L(\theta)$ คือ ฟังก์ชันความสูญเสีย (Loss Function) ของเครือข่ายประเมินค่าหลัก, $E[\dots]$ คือ ค่าคาดหวัง (Expected Value) ของสมการ, r คือ รางวัลที่ได้รับทันที, γ คือ ตัวประกอบการลดทอน, $\max_{a'}$ คือ ค่าผลลัพธ์สูงสุดเมื่อพิจารณาจากการกระทำ a' ที่เป็นไปได้ทั้งหมด, $Q(s', a'; \theta^-)$ คือ ค่า Q-value เป้าหมาย (Target Q-value) สำหรับสถานะถัดไป s' และการกระทำถัดไป a' โดยใช้พารามิเตอร์ θ^- คือ พารามิเตอร์ของเครือข่ายเป้าหมายเทคนิคนี้ทำงานร่วมกับ Replay Memory ซึ่งช่วยลดปัญหาความสัมพันธ์ของข้อมูลที่ใกล้เคียงกัน และทำให้การเรียนรู้มีเสถียรภาพมากขึ้น, $Q(s, a; \theta)$ คือ ค่า Q-value ที่คาดการณ์สำหรับสถานะปัจจุบัน s และการกระทำปัจจุบัน a โดยใช้พารามิเตอร์ θ สำหรับการนำไปใช้กับรถยนต์ไร้คนขับ DRL จะใช้ข้อมูลจากกล้องและเซ็นเซอร์ เช่น RGB-D และ LiDAR เพื่อนำมาประมวลผลผ่าน CNN ก่อนจะเข้าสู่กระบวนการตัดสินใจ โดยใช้ค่าตอบแทน (Reward) เป็นตัวชี้วัด เช่น การให้รางวัลเมื่อรักษาระยะห่างที่ปลอดภัย และให้โทษเมื่อเกิดการชน นอกจากนี้ยังใช้กลยุทธ์ Exploration-Exploitation เช่น ϵ -Greedy เพื่อสร้างสมดุลระหว่างการสำรวจเส้นทางใหม่และการใช้ข้อมูลที่เรียนรู้มาแล้วเพื่อให้ได้ประสิทธิภาพสูงสุด [11] รูปแบบการเรียนรู้แบบเสริมเชิงลึกนี้จึงเหมาะสมอย่างยิ่งสำหรับการฝึกฝน AI ที่ต้องเผชิญกับการตัดสินใจในสภาพแวดล้อมที่มีความซับซ้อนและมีมิติข้อมูลสูง โดยไม่ต้องอาศัยแบบจำลองของระบบที่ตายตัวล่วงหน้า เช่น ระบบหุ่นยนต์และการขับอัตโนมัติ ที่ตัวแทนจะต้องประมวลผลข้อมูลจากเซ็นเซอร์หลากหลายชนิดเพื่อประกอบการเคลื่อนที่ และสามารถค้นหานโยบายการกระทำที่ดีที่สุดผ่านประสบการณ์ตรงในการลองผิดลองถูกจากสภาพแวดล้อมจำลอง



รูปที่ 2.2 ตัวอย่างสถาปัตยกรรม Deep reinforcement learning

2.1.3 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning: RL)

คือส่วนหนึ่งของปัญญาประดิษฐ์ที่มุ่งเน้นการสอนให้คอมพิวเตอร์เรียนรู้ผ่านการโต้ตอบกับสภาพแวดล้อมเพื่อบรรลุเป้าหมายที่กำหนด โดยไม่ต้องอาศัยข้อมูลสอนที่มีเฉลยชัดเจนเหมือนการเรียนรู้แบบมีผู้สอน แต่จะเรียนรู้ผ่านการลองผิดลองถูก และได้รับผลป้อนกลับในรูปแบบของรางวัลและการลงโทษเพื่อปรับพฤติกรรมให้เหมาะสมที่สุด พื้นฐานทางคณิตศาสตร์ของการเรียนรู้แบบเสริมกำลัง และให้เข้าใจถึงกระบวนการตัดสินใจของ Agent ในทางคณิตศาสตร์ ปัญหาการจอร์ถอัตโนมัติสามารถอธิบายได้ด้วยกระบวนการตัดสินใจแบบมาร์คอฟ (Markov Decision Process: MDP) [8] ซึ่งประกอบด้วยเซตของตัวแปร 5 อย่าง คือ S, A, P, R, γ โดยที่ S คือ เซตของสถานะ (State) หรือภาพจากกล้องที่ตัวแทนมองเห็น, A คือ เซตของการกระทำ (Action) เช่น การเลี้ยว, การเหยียบคันเร่ง, P คือ ความน่าจะเป็นในการเปลี่ยนสถานะ, R คือ ฟังก์ชันรางวัล, γ คือ Discount Factor ที่กำหนดความสำคัญของรางวัลในอนาคต



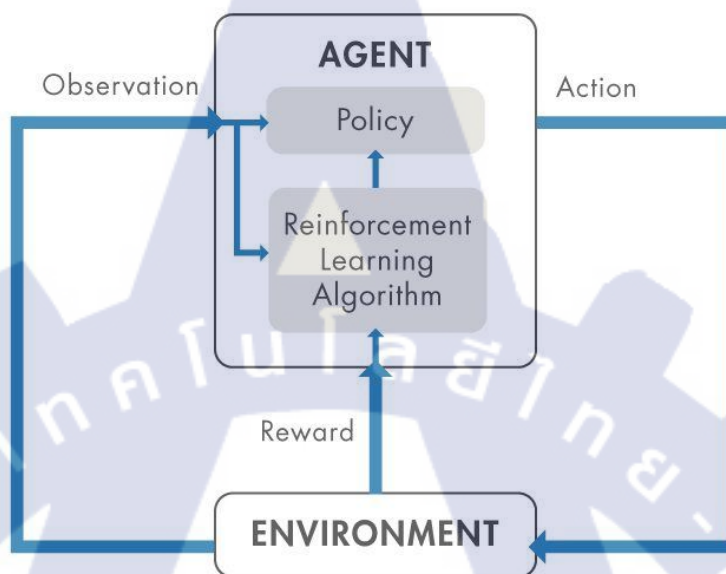
รูปที่ 2.3 ตัวอย่างของ MDP อย่างง่ายที่มีสามสถานะ (วงกลมสีเขียว) และสองการกระทำ (วงกลมสีส้ม) พร้อมสองรางวัล (ลูกศรสีส้ม)

เป้าหมายหลักของการเรียนรู้คือการค้นหานโยบายหรือ π ที่ทำให้ผลตอบแทนสะสมสูงสุด โดยความสัมพันธ์ระหว่างสถานะปัจจุบันและมูลค่าที่จะได้รับ สามารถอธิบายได้ด้วย สมการเบลล์แมน (Bellman Equation) [12] ดังสมการต่อไปนี้:

$$V^\pi(s) = E_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s] \quad (4)$$

โดยที่ $V^\pi(s)$ คือฟังก์ชันมูลค่าของสถานะ s ภายใต้นโยบาย π , E_π คือค่าคาดหวัง, R_{t+1} คือรางวัลที่ได้รับทันที, γ คือตัวประกอบการลดทอนเพื่อกำหนดความสำคัญของอนาคต, $V^\pi(S_{t+1})$ คือ ฟังก์ชันมูลค่าประเมินของสถานะถัดไป S_{t+1} ภายใต้นโยบาย π , S_{t+1} คือ สถานะถัดไป โดยมีเงื่อนไขว่าสถานะเริ่มต้น ณ เวลา t คือ s ($S_t = s$) สมการนี้มีความหมายว่า มูลค่าของสถานะหนึ่งๆ เท่ากับ รางวัลที่ได้รับทันที บวกกับ มูลค่าประเมินของสถานะถัดไป ซึ่งเป็น หัวใจสำคัญที่ทำให้ AI สามารถวางแผนล่วงหน้าได้ ไม่ใช่แค่ตัดสินใจเฉพาะหน้าเท่านั้น ดังนั้น การเรียนรู้แบบเสริมกำลังจึงมีความเหมาะสมเป็นอย่างมากสำหรับการฝึกฝน AI ในงานที่เน้นกระบวนการตัดสินใจอย่างเป็นลำดับขั้นตอนเพื่อบรรลุเป้าหมายระยะยาว โดยไม่ต้องอาศัยชุดข้อมูลที่มีเฉลยคำตอบล่วงหน้า ซึ่งตอบโจทย์อย่างยิ่งกับระบบที่สามารถสร้างแบบจำลองสภาพแวดล้อมแบบ MDP

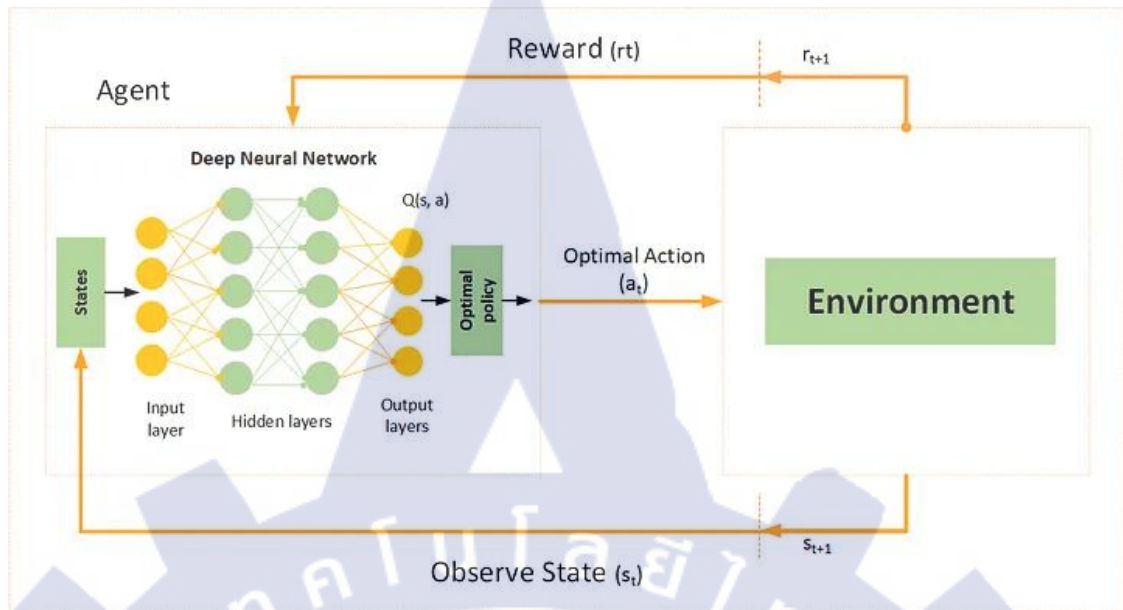
และระบบควบคุมอัตโนมัติที่เน้นการปรับปรุงพฤติกรรมผ่านผลป้อนกลับ เช่น การจราจรอัตโนมัติ ที่ AI จะต้องวางแผนล่วงหน้าเพื่อกำหนดนโยบายที่ให้ผลตอบแทนสะสมสูงสุด



รูปที่ 2.4 ตัวอย่างสถาปัตยกรรม Reinforcement Learning

2.1.4 Deep Q Network (DQN)

คือ อัลกอริทึมการเรียนรู้แบบเสริมเชิงลึก เพื่อนำมาใช้แทน Q-Learning แบบดั้งเดิมซึ่งมีข้อจำกัดในการจัดการกับปัญหาที่มี State Space ขนาดใหญ่โดยใช้โครง Deep Neural Networks ในการประมาณค่าของฟังก์ชัน Q แทนการใช้ Q-Table [13] ที่มีขนาดใหญ่เกินกว่าจะจัดเก็บได้อย่างมีประสิทธิภาพ โครงสร้างของ DQN ประกอบด้วยองค์ประกอบสำคัญ ได้แก่ Replay Memory ซึ่งเป็นการเก็บประสบการณ์การโต้ตอบของตัวแทน กับสภาพแวดล้อมไว้ในบัฟเฟอร์และเรียกใช้แบบสุ่มระหว่างการฝึกเพื่อช่วยลดปัญหาความสัมพันธ์ของข้อมูล Target Network ซึ่งเป็นโครงข่ายประสาทเทียมชุดที่สองที่คัดลอกค่าจากโครงข่ายหลัก ทุก ๆ C ขั้นตอนเพื่อลดความไม่เสถียรของการเรียนรู้ และ Exploration-Exploitation Strategy [14] เช่น ϵ -Greedy ที่ใช้กำหนดโอกาสในการเลือกการกระทำแบบสุ่มเพื่อสำรวจหรือเลือกการกระทำที่ให้ค่าตอบแทนสูงสุดค่าของฟังก์ชัน Q



รูปที่ 2.5 ตัวอย่างสถาปัตยกรรมของ Deep Q Network

2.1.5 Convolutional Neural Network (CNN)

คือ รูปแบบหนึ่งของโครงข่ายประสาทเทียมเชิงลึกที่ออกแบบมาเพื่อวิเคราะห์และประมวลผลข้อมูลเชิงภาพโดยเฉพาะ โดยสามารถเรียนรู้คุณลักษณะสำคัญ จากภาพได้โดยอัตโนมัติ โดยไม่ต้องพึ่งพาการกำหนดพิกเจอร์ด้วยมือ เหมือนวิธีการดั้งเดิม เช่น Histogram of Oriented Gradients (HOG) [15] หรือ Support Vector Machines (SVM) [16] CNN ได้ถูกนำไปใช้ในหลากหลายแอปพลิเคชัน เช่น การจดจำใบหน้า การตรวจจับวัตถุ และการขับขี่อัตโนมัติ โดยองค์ประกอบของ CNN ประกอบด้วย 3 ส่วนหลัก ได้แก่ Convolutional Layer, Pooling Layer และ Fully Connected Layer ซึ่งแต่ละส่วนมีหน้าที่และการคำนวณทางคณิตศาสตร์ที่สำคัญดังนี้

1. Convolutional Layer เป็นส่วนหัวใจสำคัญของ CNN ทำหน้าที่สกัดคุณลักษณะจากข้อมูลอินพุตผ่านตัวกรองที่เรียกว่า Kernel หรือ Filter กระบวนการทำงานเริ่มจากการเลื่อนตัวกรองไปตามภาพอินพุต จากนั้นคำนวณผลรวมของจุดตัดระหว่างค่าพิกเซลของภาพกับค่าของตัวกรอง ซึ่งสามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์สำหรับการคำนวณค่า Feature Map (Z) ณ ตำแหน่ง (i, j) ดังนี้:

$$Z_{i,j} = \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} X_{i+u,j+v} \cdot K_{u,v} + b \quad (5)$$

โดยที่ $Z_{i,j}$ คือ แผนที่ฟีเจอร์ (Feature Map) ผลลัพธ์ที่ตำแหน่ง i, j , $X_{i+u, j+v}$ คือ ค่าพิกเซลของเมทริกซ์ภาพอินพุต (Input Image Matrix) ณ ตำแหน่ง $i + u, j + v$, $K_{u,v}$ คือ ค่าของเมทริกซ์ตัวกรอง (Kernel/Filter Matrix) ณ ตำแหน่ง u, v , ซึ่งแสดงถึงคุณลักษณะเฉพาะที่ถูกสกัดออกมา เช่น ขอบภาพ เส้น หรือลวดลาย, H คือ ขนาดความสูงของตัวกรอง, W คือ ขนาดความกว้าง ของตัวกรอง, b คือ ค่าไบอัส, i, j คือ พิกัดตำแหน่งพิกเซลของ Feature Map ผลลัพธ์, u, v คือ ตัวแปรดัชนี (Index) สำหรับการวนรอบขนาดของตัวกรอง

2. Pooling Layer เป็นชั้นที่ทำหน้าที่ลดขนาดมิติของแผนที่ฟีเจอร์ เพื่อลดจำนวนพารามิเตอร์และการคำนวณในโมเดล รวมถึงช่วยป้องกันภาวะ Overfitting โดยการเลือกตัวแทนข้อมูลในแต่ละบริเวณ R มี 2 ประเภทหลัก ได้แก่ Max Pooling คือการเลือกค่า pixel ที่มีค่ามากที่สุด ในบริเวณนั้น ดังสมการ:

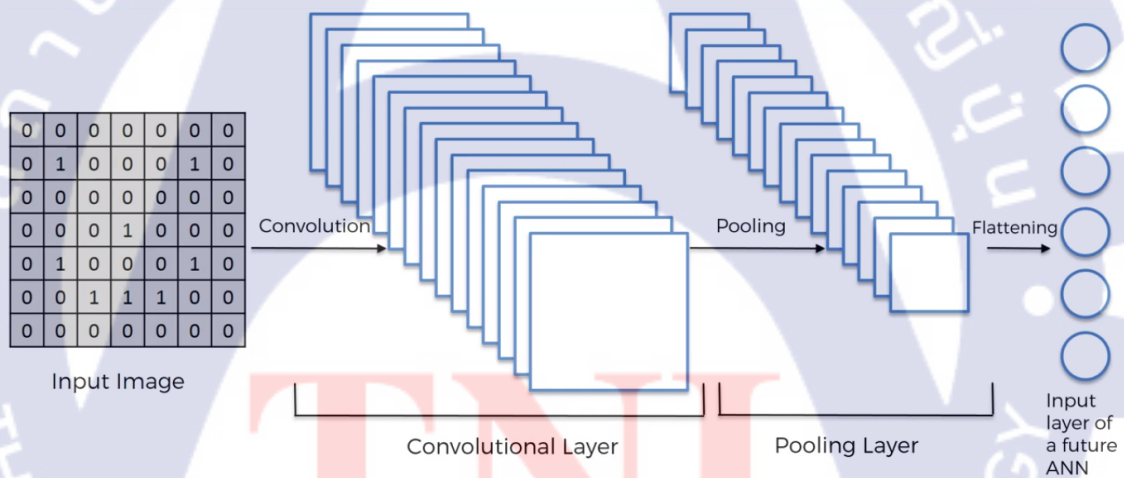
$$Y_{max} = \max_{(p,q) \in R_{i,j}} (z_{p,q}) \quad (6)$$

โดยที่ Y_{max} คือ ค่าผลลัพธ์พิกเซลที่มีค่ามากที่สุดหลังจากการทำ Max Pooling, $\max$ คือ ฟังก์ชันคณิตศาสตร์สำหรับหาค่าสูงสุด, $z_{p,q}$ คือ ค่าของพิกเซลบนแผนที่ฟีเจอร์ ณ ตำแหน่ง (p, q) , $R_{i,j}$ คือ บริเวณของ Pooling Window ณ ตำแหน่ง i, j บนแผนที่ฟีเจอร์, (p, q) คือ พิกัดตำแหน่งที่อยู่ภายในบริเวณ $R_{i,j}$, i, j คือ พิกัดตำแหน่งของข้อมูลผลลัพธ์ใน เลเยอร์ ปัจจุบัน

3. Fully Connected Layer ขั้นสุดท้ายทำหน้าที่จำแนกประเภทของข้อมูลที่ได้จากขั้นตอนก่อนหน้า โดยเริ่มจากการปรับข้อมูลแผนที่ฟีเจอร์ให้เป็นเวกเตอร์ 1 มิติ และส่งเข้าสู่โครงข่ายประสาทเทียมแบบดั้งเดิม เพื่อคำนวณค่าความน่าจะเป็นของแต่ละคลาส โดยมักใช้ฟังก์ชัน Softmax ในเลเยอร์สุดท้าย ดังสมการ:

$$P(y = j|x) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (7)$$

โดยที่ $P(y = j|x)$ คือ ค่าความน่าจะเป็น (Probability) ที่ข้อมูลอินพุต x จะจัดอยู่ในประเภท j , e คือ ค่าคงที่ทางคณิตศาสตร์ (Euler's Number), z_j คือ ค่าเอาต์พุตดิบ (Logit) ในเลเยอร์สุดท้ายสำหรับประเภท j , z_k คือ ค่าเอาต์พุตดิบ (Logit) ในเลเยอร์สุดท้ายสำหรับประเภท k , K คือ จำนวนประเภทหรือคลาสทั้งหมด (Total Classes), j คือ ตัวแปรแทนประเภทเป้าหมายที่ต้องการหาความน่าจะเป็น, k คือ ตัวแปรดัชนี (Index) สำหรับนำมารวมค่าของประเภททั้งหมด, x คือ ข้อมูลอินพุตและ y คือ ตัวแปรสุ่มแทนผลลัพธ์ประเภทข้อมูล ซึ่งจะให้ผลลัพธ์เป็นค่าความน่าจะเป็นที่วัตถุในภาพจะจัดอยู่ในประเภท j จากจำนวนประเภททั้งหมด K ข้อดีของ CNN คือสามารถเรียนรู้ฟีเจอร์จากข้อมูลภาพได้โดยอัตโนมัติและมีประสิทธิภาพสูงกว่าวิธีดั้งเดิม สามารถรองรับข้อมูลขนาดใหญ่ที่มีความซับซ้อนได้ดี แต่มีข้อเสียคือต้องใช้ทรัพยากรการประมวลผลสูง โดยเฉพาะเมื่อโครงข่ายมีความลึกมากและต้องการข้อมูลจำนวนมาก ในการฝึกฝนเพื่อป้องกันปัญหา Overfitting จึงอาจจำเป็นต้องมีการปรับโครงสร้างโมเดลและใช้เทคนิค Data Augmentation ให้เหมาะสมกับงานแต่ละประเภท [17]



รูปที่ 2.6 Convolutional Layer และ การทำ Flattening

2.1.6 Proximal Policy Optimization (PPO)

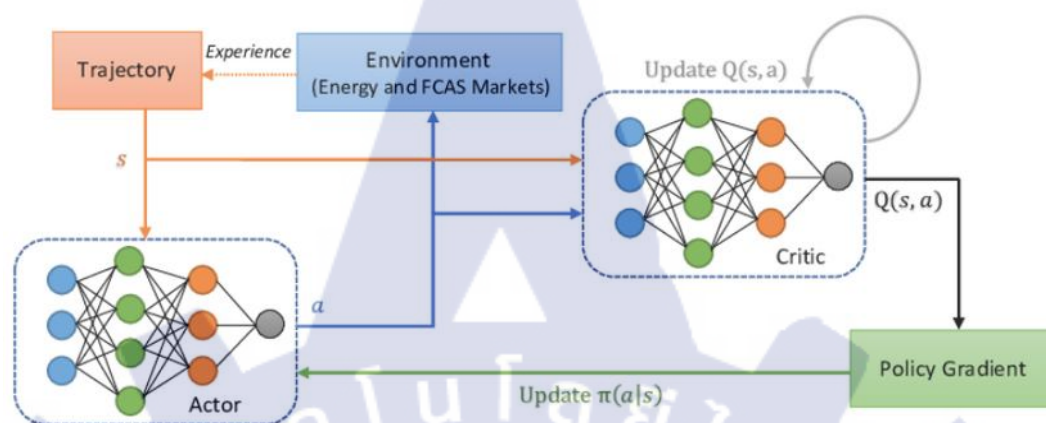
คือ รูปแบบหนึ่งของอัลกอริทึมการเรียนรู้แบบเสริมกำลัง (RL) ประเภท Policy Gradient ที่ออกแบบมาเพื่อแก้ปัญหาค่าความคาดหวังและการตัดสินใจในพื้นที่การกระทำที่ต่อเนื่องโดยเฉพาะ ซึ่งสามารถเรียนรู้นโยบายการตัดสินใจที่เหมาะสมที่สุดได้โดยอัตโนมัติ และมีความเสถียรในการฝึกฝนสูงกว่าวิธีการดั้งเดิม เช่น DQN ที่มีปัญหาความผันผวน หรือ Trust Region Policy Optimization (TRPO) [18] ที่มีความซับซ้อนในการคำนวณสูง PPO ได้ถูกนำไปใช้ในหลากหลายแอปพลิเคชัน

ระดับสูง เช่น การควบคุมหุ่นยนต์, การเล่นเกมกลยุทธ์, และการขับซื้ออัตโนมัติ โดยองค์ประกอบและกระบวนการทำงานของ PPO ประกอบด้วย 3 ส่วนหลัก ได้แก่ Actor (Policy Network), Critic (Value Network), และ Clipped Surrogate Objective แต่ละส่วนมีหน้าที่สำคัญในการประมวลผลและปรับปรุงการเรียนรู้ ได้แก่ Actor เป็นส่วนสำคัญที่ทำหน้าที่เปรียบเสมือน "ผู้กระทำ" โดยรับข้อมูลสถานะ จากสิ่งแวดล้อมและทำการสุ่มเลือกการกระทำ ตามการแจกแจงความน่าจะเป็นที่เรียนรู้มา เป้าหมายของ Actor คือการค้นหานโยบายที่ให้ผลตอบแทนสูงสุด กระบวนการนี้คล้ายกับการสกัดคุณลักษณะใน CNN แต่เปลี่ยนจากการหาพีเจอร์เป็นการหา "การกระทำที่ดีที่สุด" ในสถานการณ์นั้นๆ ต่อไปคือ Critic เป็นส่วนที่ทำหน้าที่เปรียบเสมือน ผู้วิจารณ์ที่คอยประเมินว่าสถานะปัจจุบันที่ Agent อยู่มีความดีหรือแย่เพียงใด โดยการคำนวณค่าคาดการณ์ผลตอบแทนสะสม ข้อมูลจาก Critic จะถูกนำไปใช้คำนวณค่าความได้เปรียบ เพื่อบอกว่าการกระทำที่ Actor เลือกนั้น ดีกว่าค่าเฉลี่ยปกติมากน้อยเพียงใด ซึ่งช่วยลดความแปรปรวน ในการฝึกฝน ส่วนสุดท้ายคือ Clipped Surrogate Objective เป็นกลไกหัวใจสำคัญของ PPO ที่ทำหน้าที่ควบคุมการอัปเดตนโยบายไม่ให้เปลี่ยนแปลงไปอย่างรวดเร็วเกินไป คล้ายกับชั้น Pooling ที่คัดกรองข้อมูล แต่ในที่นี้คือการ Clip ค่าความน่าจะเป็นของการกระทำให้อยู่ในช่วงที่กำหนด หากการอัปเดตนั้นส่งผลให้พารามิเตอร์เปลี่ยนไปมากเกินไป ระบบจะทำการตัดค่าส่วนเกินทิ้ง เพื่อป้องกันไม่ให้ประสิทธิภาพของโมเดลตกลงอย่างกะทันหัน ซึ่งสามารถเขียนเป็นสมการทางคณิตศาสตร์ได้ดังนี้

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (8)$$

โดยที่ $L^{CLIP}(\theta)$ คือ ฟังก์ชันวัตถุประสงค์แบบมีการตัดทอนของ PPO, θ คือ ค่าพารามิเตอร์ปัจจุบันของโครงข่ายนโยบาย, $\hat{E}_t$ คือ ค่าคาดหวังเชิงประจักษ์ (Empirical Expectation) ณ เวลา t , $\min$ คือ ฟังก์ชันคณิตศาสตร์สำหรับเลือกค่าที่ต่ำที่สุดระหว่างสองเงื่อนไข, $r_t(\theta)$ คือ อัตราส่วนความน่าจะเป็นระหว่างนโยบายใหม่กับนโยบายเก่า (Probability Ratio), $\hat{A}_t$ คือ ฟังก์ชันความได้เปรียบประเมิน ณ เวลา t , clip คือ ฟังก์ชันการจำกัดขอบเขตไม่ให้ค่าเกินกว่าช่วงที่กำหนด, ϵ คือ ค่าคงที่ไฮเปอร์พารามิเตอร์ สำหรับกำหนดขอบเขตการเปลี่ยนแปลงนโยบายและ t คือ ลำดับเวลาปัจจุบัน ข้อดีของ PPO คือ มีความเสถียรสูงและปรับปรุงพารามิเตอร์ได้ง่ายกว่าอัลกอริทึมรุ่นก่อนหน้า สามารถรองรับการกระทำแบบต่อเนื่องได้ดีเยี่ยม ซึ่งเหมาะกับงานควบคุมหุ่นยนต์ และมีสมดุลที่ระหว่างความง่ายในการนำไปใช้ และประสิทธิภาพ แต่ข้อเสียคือมีใช้ข้อมูลสิ้นเปลืองกว่าอัลกอริทึมแบบ Off-policy บางประเภท [19] เนื่องจากต้องเก็บ

ข้อมูลใหม่ทุกครั้งที่มีการอัปเดตนโยบาย และต้องอาศัยการกำหนด Reward Function ที่รัดกุมเพื่อให้ได้พฤติกรรมที่ต้องการ



รูปที่ 2.7 ตัวอย่างสถาปัตยกรรม Proximal Policy Optimization

2.1.7 Game Engine

คือ ซอฟต์แวร์ framework ที่รวบรวมเครื่องมือและระบบที่จำเป็นสำหรับการพัฒนาเกม และการจำลองสถานการณ์ไว้ในที่ ๆ เดียว ช่วยให้นักพัฒนาสามารถสร้างและจัดการองค์ประกอบต่าง ๆ ได้อย่างมีประสิทธิภาพโดยไม่ต้องเขียนโค้ด Low-level code ขึ้นมาใหม่ทั้งหมด องค์ประกอบหลัก ประกอบด้วย กราฟิกเรนเดอร์เอนจินสำหรับภาพ 2 มิติ และ 3 มิติ, ระบบจำลองฟิสิกส์, ระบบเสียง, AI, การจัดการอินพุต, และระบบเครือข่าย ในปัจจุบัน Game Engine ที่ได้รับความนิยมในวงการอุตสาหกรรมและการวิจัย ได้แก่ Unity, Unreal Engine และ Godot ซึ่งแต่ละตัวมีจุดเด่นที่แตกต่าง กัน กล่าวคือ Unity มีความยืดหยุ่นสูงรองรับหลายแพลตฟอร์มและมีชุมชนผู้ใช้ขนาดใหญ่, Unreal Engine โดดเด่นด้านคุณภาพกราฟิกที่สมจริงระดับสูง, และ Godot เป็นซอฟต์แวร์โอเพนซอร์สที่เน้น ความเบาและรองรับหลายภาษาโปรแกรม ในบริบทของงานวิจัยด้านปัญญาประดิษฐ์ Game Engine ได้ถูกนำมาใช้อย่างแพร่หลายในการพัฒนาและฝึกฝนโมเดลผ่านรูปแบบ DRL ผ่านการจำลอง สภาพแวดล้อมเสมือนจริง ซึ่งช่วยให้การทดสอบและพัฒนา AI มีความแม่นยำ ปลอดภัย และมี ประสิทธิภาพสูงกว่าการทดสอบในโลกจริง โดยเฉพาะในงานที่ต้องอาศัยการลองผิดลองถูกที่มีความ เสี่ยงสูง เช่น การขับขี้อัตโนมัติ สำหรับการศึกษา นักวิจัยเลือกใช้ Unity Game Engine เวอร์ชัน 2022.3.61f1 เนื่องจากเป็นแพลตฟอร์มที่มีความเหมาะสมสำหรับการวิจัย Intelligent Agents โดย ทำงานร่วมกับชุดเครื่องมือ Unity ML-Agents Toolkit [20], [21], [22] ซึ่งช่วยให้ Unity สามารถ เชื่อมต่อกับเฟรมเวิร์ก Python API ภายนอกผ่านโปรโตคอลสื่อสาร (Remote Procedure Calls)

gRPC [23] เพื่อรับส่งข้อมูลการสังเกตการณ์ จากเซนเซอร์และคำสั่งควบคุม ได้อย่างรวดเร็ว
ความสามารถที่สำคัญของ Unity 2022.3.61f1 ที่สนับสนุนงานวิจัยนี้ประกอบด้วย:

1. ความซับซ้อนทางกายภาพ: รองรับการจำลองฟิสิกส์ผ่าน Nvidia PhysX [24] ทำให้สามารถจำลองพลวัตของยานพาหนะ แรงเสียดทาน และการชนได้อย่างสมจริง ซึ่งเป็นปัจจัยสำคัญในการฝึกฝนตัวแทนให้เรียนรู้นโยบายการขับขี่ที่ปลอดภัย
2. ความซับซ้อนทางประสาทสัมผัส: สามารถจำลองข้อมูลนำเข้าได้หลากหลายรูปแบบ ทั้งข้อมูลภาพ จากกล้องเสมือนที่มีความสมจริง และข้อมูลระยะทางที่จำลองการทำงานของเซนเซอร์ LiDAR
3. ประสิทธิภาพการฝึกฝน: ระบบรองรับการเร่งความเร็วการจำลอง ได้สูงกว่าเวลาจริงถึง 100 เท่า และรองรับการฝึกฝนแบบขนาน หลายตัวแทนพร้อมกัน ซึ่งช่วยลดระยะเวลาในการเก็บข้อมูลตัวอย่างสำหรับอัลกอริทึม PPO หรือ SAC ได้อย่างมาก
ด้วยคุณสมบัติเหล่านี้ของ Unity 2022.3.61f1 จึงทำหน้าที่เป็นแพลตฟอร์มกลาง ที่ช่วยลดช่องว่างระหว่างการจำลองและการใช้งานจริง และช่วยในการพัฒนาระบบขับขี่อัตโนมัติที่มีความฉลาดและปลอดภัยยิ่งขึ้น



รูปที่ 2.8 โปรแกรม Unity สำหรับสร้างสภาพแวดล้อมจำลอง

2.2 ทบทวนวรรณกรรม

2.2.1 Improving parking availability prediction in smart cities with IoT and ensemble-based model [1]

งานวิจัยนี้ออกแบบและพัฒนาระบบทำนายความพร้อมใช้งานของที่จอดรถในเมืองอัจฉริยะโดยบูรณาการเทคโนโลยี IoT ร่วมกับโมเดลการเรียนรู้ของเครื่องแบบกลุ่ม สำหรับวิเคราะห์และทำนายข้อมูลแทนการใช้การประมวลผลภาพแบบเดิมที่มีความซับซ้อนสูง โดยมีการเปรียบเทียบอัลกอริทึมกลุ่ม Bagging และ Boosting ได้แก่ Random Forest Regressor, Bagging Regression, Adaptive Boosting และ Gradient Boosting เพื่อค้นหาโมเดลที่แม่นยำที่สุด แม้ว่าจะงานวิจัยก่อนหน้านี้จะนิยมใช้ Deep Learning

หรือ CNN ในการตรวจจับภาพ แต่แนวทางนี้เน้นการจัดการข้อมูลตัวเลข จากเซนเซอร์เพื่อลดความซับซ้อน และเพิ่มประสิทธิภาพ ข้อมูลจากเซนเซอร์จะถูกส่งผ่านเกตเวย์ไปยังระบบคลาวด์เพื่อประมวลผล โดยผลการทดลองกับชุดข้อมูลที่จอดรถเมืองเบอร์มิงแฮมแสดงให้เห็นว่าอัลกอริทึม Bagging Regression (BR) ให้ประสิทธิภาพสูงสุดด้วยค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ย (MAE) เพียง 0.06% ซึ่งปรับปรุงประสิทธิภาพให้ดีกว่าวิธีที่มีอยู่เดิมกว่า 6.6% พร้อมทั้งช่วยลดความซับซ้อนของระบบได้อย่างมีนัยสำคัญ ซึ่งสามารถนำไปประยุกต์ใช้เพื่อลดปัญหาการจราจรติดขัดจากการวนหาที่จอดรถในเขตเมืองได้อย่างมีประสิทธิภาพ

2.2.2 IoT Based Smart Parking System Using Deep Long Short Memory Network [2]

งานวิจัยนี้ออกแบบและพัฒนาระบบการทำงานระบบจอดรถอัจฉริยะโดยบูรณาการเทคโนโลยี IoT และเครือข่ายเซนเซอร์ผ่าน Deep Neural Network แบบ Deep Long Short Term Memory (Deep LSTM) สำหรับทำนายความพร้อมใช้งานของพื้นที่จอดรถซึ่งมีลักษณะข้อมูลเป็นอนุกรมเวลา (Time-series) โดยใช้โมเดล Deep LSTM ที่มีความโดดเด่นในการเรียนรู้ความสัมพันธ์ระยะยาวและเหมาะสมกับการจัดการข้อมูลที่มีความขึ้นกับเวลามากกว่าโครงข่ายประสาทเทียมแบบ Feed-forward ทั่วไป ข้อมูลสถานะการจอดรถจะถูกรวบรวมจากเซนเซอร์และส่งผ่านโปรโตคอลการสื่อสาร LoRaWAN และ MQTT ไปยังระบบคลาวด์เพื่อประมวลผลข้อมูลขนาดใหญ่และสนับสนุนการตัดสินใจ โดยผลการทดลองกับชุดข้อมูลเมืองเบอร์มิงแฮมแสดงให้เห็นว่าระบบสามารถทำนายจำนวนที่จอดรถว่างได้อย่างมีประสิทธิภาพทั้งในมิติของสถานที่ วัน และเวลา โดยให้ค่าความคลาดเคลื่อนต่ำกว่าเทคนิคดั้งเดิมอย่าง Polynomials, Fourier Series หรือ RNN ธรรมดา ซึ่งสามารถนำไปประยุกต์ใช้เพื่อลดระยะเวลาในการค้นหาที่จอดรถและลดปัญหาการจราจรติดขัดในเขตเมืองได้อย่างเป็นรูปธรรม

2.2.3 Daimler and Bosch's driverless parking gets OK to operate without human supervision [3]

งานพัฒนานี้ออกแบบและทดสอบระบบบริการนำรถไปจอดอัตโนมัติ โดยบูรณาการเทคโนโลยียานยนต์ของ Daimler เข้ากับโครงสร้างพื้นฐานอัจฉริยะของ Bosch เพื่อให้รถยนต์สามารถขับเคลื่อนและเข้าจอดได้เองโดยสมบูรณ์โดยไม่ต้องมีมนุษย์ควบคุม ซึ่งจัดเป็นเทคโนโลยีขับเคลื่อนอัตโนมัติระดับ 4 ที่ได้รับการอนุมัติให้ใช้งานจริงในชีวิตประจำวันครั้งแรกในโลกโดยหน่วยงานกำกับดูแลของเยอรมนี ระบบทำงานโดยเชื่อมโยงคำสั่งจองผ่านแอปพลิเคชันสมาร์ทโฟน รถยนต์จะเคลื่อนที่จากจุดรับส่งไปยังช่องจอดที่กำหนดโดยอาศัยการนำทางจากการประมวลผลข้อมูลร่วมกันระหว่างเซนเซอร์บนตัวรถและอุปกรณ์โครงสร้างพื้นฐานที่ติดตั้งภายในอาคารจอดรถ นอกจากนี้ยังมีการออกแบบระบบสัญญาณไฟสีเทอร์ควอยซ์เพื่อระบุสถานะการขับเคลื่อนอัตโนมัติให้ผู้สัญจรรอบข้างรับทราบตามมาตรฐาน SAE 3134 ผลการพัฒนาแสดงให้เห็นว่าระบบมีความปลอดภัยและประสิทธิภาพสูงเพียงพอที่จะปฏิบัติการโดยปราศจากคนขับสำรอง นั่ง

หลังพวงมาลัย ซึ่งถือเป็นต้นแบบสำคัญในการกำหนดมาตรฐานและกฎระเบียบสำหรับการสัญจรและระบบจราจรไร้คนขับในอนาคต

2.2.4 Smart Parking Lot Based on Edge Cluster Computing for Full Self-Driving Vehicles [4]

งานวิจัยนี้ออกแบบและพัฒนาระบบลานจอดรถอัจฉริยะสำหรับรถยนต์ขับเคลื่อนอัตโนมัติเต็มรูปแบบโดยใช้เทคโนโลยี Edge Cluster Computing เพื่อแก้ปัญหาข้อจำกัดด้านการสื่อสารและทรัพยากรการประมวลผลของรถยนต์ในสภาพแวดล้อมปิดอย่างอาคารจอดรถ โดยแบ่งโครงสร้างการทำงานออกเป็น Edge Terminal ที่ติดตั้งคงที่ซึ่งใช้ฮาร์ดแวร์ Raspberry Pi และ Mobile Edge ซึ่งคือตัวรถยนต์ ทำงานร่วมกันผ่านแผนที่แบบตาราง สำหรับการจัดการพื้นที่ การระบุตำแหน่ง และการนำทาง ระบบใช้อัลกอริทึมการเชื่อมต่อแบบกลุ่ม ที่ให้อุปกรณ์ Edge สามารถสื่อสารและเปลี่ยนสถานะระหว่าง Follower และ Spanner เพื่อกระจายคำสั่งจองและค้นหาช่องจอดว่างครอบคลุมทั่วทั้งพื้นที่จอดรถแทนการประมวลผลแบบเดี่ยว การทดสอบระบบดำเนินการผ่านโปรแกรมจำลอง Raspberry Pi Simulator (RaSim) โดยผลการทดลองแสดงให้เห็นว่าการจัดวาง Edge Terminal จำนวน 13 จุดอย่างเหมาะสม สามารถลดระยะทางและเวลาในการเคลื่อนที่เพื่อค้นหาที่จอดรถได้ถึง 47% เมื่อเทียบกับลานจอดรถแบบทั่วไป ซึ่งพิสูจน์ได้ว่าระบบนี้ช่วยเพิ่มประสิทธิภาพการบริหารจัดการพลังงานและลดความล่าช้าในการเข้าจอดในพื้นที่ที่การเชื่อมต่อกับเครือข่ายภายนอกทำได้ยาก

2.2.5 Review on the Safety and Sustainability of Autonomous Vehicles: Challenges and Future Directions [5]

งานวิจัยนี้วิเคราะห์และนำเสนอแนวทางการพัฒนายานยนต์ขับเคลื่อนอัตโนมัติโดยเน้นความปลอดภัยและความยั่งยืนผ่านการประมวลผลข้อมูลจาก Sensor Fusion และปัญญาประดิษฐ์ผ่าน Deep Neural Network สำหรับตรวจจับและจำแนกวัตถุรวมถึงคนเดินเท้าในสภาพแวดล้อมที่มีพลวัต โดยเปรียบเทียบการใช้โมเดล YOLO ที่โดดเด่นด้านความเร็วในการแบ่งส่วนภาพและทำนายกรอบวัตถุซึ่งเหมาะสมกับการทำงานแบบเรียลไทม์ กับโมเดล CNN ที่ให้ความแม่นยำสูงในการสกัดคุณลักษณะที่ซับซ้อนแม้อาจใช้ทรัพยากรการประมวลผลมากกว่า ส่วนการระบุตำแหน่งและการนำทางใช้เทคนิคการผสานข้อมูลจาก LiDAR, Radar และกล้องร่วมกับแผนที่ความละเอียดสูงและ GPS เพื่อลดข้อจำกัดของเซนเซอร์ในสภาวะอากาศเลวร้าย ข้อมูลการขับขี่และการจราจรจะถูกเชื่อมโยงผ่านระบบสื่อสาร V2X บนเครือข่าย 5G ซึ่งช่วยให้ยานพาหนะแลกเปลี่ยนข้อมูลกับโครงสร้างพื้นฐานได้อย่างรวดเร็ว โดยผลการศึกษาชี้ให้เห็นว่าการบูรณาการอัลกอริทึมการตัดสินใจทางจริยธรรมเข้ากับระบบตรวจจับที่มีประสิทธิภาพ

สามารถลดความเสี่ยงจากอุบัติเหตุและเพิ่มประสิทธิภาพการใช้พลังงาน ซึ่งเป็นแนวทางสำคัญในการกำหนดมาตรฐานกฎหมายและการสร้างความยอมรับของสังคมต่อระบบขนส่งอัจฉริยะในอนาคต

2.2.6 Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses [6]

งานวิจัยนี้สำรวจและวิเคราะห์ความมั่นคงปลอดภัยของระบบยานยนต์ขับเคลื่อนอัตโนมัติที่ประมวลผลด้วยเทคโนโลยี Deep Learning โดยเน้นกระบวนการทำงานของโครงข่ายประสาทเทียมในขั้นการรับรู้ สำหรับตรวจจับและจำแนกวัตถุผ่านข้อมูลจากเซนเซอร์หลากชนิด เช่น กล้อง, LiDAR และเรดาร์ โดยมีการพิจารณาโมเดลที่มีประสิทธิภาพสูงอย่าง YOLO และ Faster R-CNN สำหรับการตรวจจับภาพ รวมถึง Voxel Net สำหรับประมวลผลข้อมูล Point Cloud แม้เทคโนโลยีเหล่านี้จะช่วยในการวางแผนเส้นทางและการตัดสินใจแบบเรียลไทม์ แต่ผลการศึกษาชี้ให้เห็นถึงความเสี่ยงต่อการโจมตีทั้งทางกายภาพ เช่น การรบกวนสัญญาณเซนเซอร์หรือการปลอมแปลงข้อมูล และการโจมตีเชิงประตัก ซึ่งสร้างสัญญาณรบกวนในภาพเพื่อหลอกโมเดล AI ข้อมูลความเสี่ยงเหล่านี้ถูกนำมาสังเคราะห์เพื่อเสนอแนวทางการป้องกันเชิงรุกและเชิงรับ เช่น การฝึกฝนโมเดลด้วยตัวอย่างประตัก การสร้างความซับซ้อนของฮาร์ดแวร์และการตรวจจับความผิดปกติผ่านระบบคลาวด์ เพื่อยกระดับความเสถียรและความปลอดภัยของระบบก่อนนำไปประยุกต์ใช้ในอุตสาหกรรมยานยนต์จริง

2.2.7 Computing Systems for Autonomous Driving: State of the Art and Challenges [7]

งานวิจัยนี้วิเคราะห์และนำเสนอสถาปัตยกรรมระบบคอมพิวเตอร์สำหรับรถยนต์ขับเคลื่อนอัตโนมัติโดยเน้นการบูรณาการเทคโนโลยีเซนเซอร์หลากชนิด เช่น กล้อง LiDAR และ Radar เข้ากับการประมวลผลด้วย Deep Learning เพื่อการรับรู้สภาพแวดล้อมและการตัดสินใจแบบเรียลไทม์ โดยมีการเปรียบเทียบโมเดลตรวจจับวัตถุระหว่างตระกูล R-CNN ที่มีความแม่นยำสูงกับโมเดลตระกูล YOLO ซึ่งให้ประสิทธิภาพด้านความเร็วที่เหนือกว่าและเหมาะสมกับการใช้งานจริงบนท้องถนน ส่วนการตรวจจับข้อบกพร่องได้พัฒนาจากวิธีการดั้งเดิมอย่าง Hough Transform มาเป็นการใช้โครงข่ายประสาทเทียมแบบ CNN และ LSTM เพื่อแก้ไขข้อจำกัดในการตรวจจับเส้นทางที่มีความโค้งหรือโครงสร้างซับซ้อน ข้อมูลปริมาณมหาศาลจากเซนเซอร์จะถูกประมวลผลผ่านฮาร์ดแวร์เฉพาะทาง เช่น NVIDIA Drive AGX หรือ FPGAs ที่รองรับการทำงานของระบบปฏิบัติการแบบเรียลไทม์ และ Middleware อย่าง ROS โดยบทสรุปชี้ให้เห็นว่าแม้เทคโนโลยีปัจจุบันจะก้าวหน้าแต่ยังคงเผชิญความท้าทายสำคัญด้านความปลอดภัยในสภาพอากาศเลวร้าย การบริโภคพลังงาน และความเสถียรของระบบ ซึ่งจำเป็นต้องมีการออกแบบร่วมกันระหว่างฮาร์ดแวร์และซอฟต์แวร์ที่รัดกุมเพื่อให้บรรลุเป้าหมายการขับเคลื่อนอัตโนมัติระดับ 5 อย่างสมบูรณ์

2.2.8 Deep Reinforcement Learning for Autonomous Driving: A Survey [8]

งานวิจัยนี้สำรวจและวิเคราะห์การประยุกต์ใช้ DRL สำหรับพัฒนาระบบการตัดสินใจและการควบคุมของรถยนต์ขับเคลื่อนอัตโนมัติ โดยมุ่งเน้นการเรียนรู้นโยบายการขับขี่ ในสภาพแวดล้อมที่มีความซับซ้อนและมีมิติสูงผ่านอัลกอริทึมที่มีประสิทธิภาพอย่าง DQN และ Actor-Critic ซึ่งมีความยืดหยุ่นในการจัดการกับความไม่แน่นอนของสภาพแวดล้อมได้ดีกว่าวิธีการควบคุมแบบดั้งเดิมอย่าง PID หรือ Model Predictive Control (MPC) ที่มักต้องอาศัยแบบจำลองทางคณิตศาสตร์ที่ตายตัว ส่วนกระบวนการฝึกฝนโมเดลใช้การจำลองผ่านซอฟต์แวร์ความเที่ยงตรงสูง เช่น CARLA และ TORCS เพื่อประมวลผลข้อมูลสถานะจากเซนเซอร์กล้องและ LiDAR โดยมีเป้าหมายเพื่อลดความเสี่ยงและช่องว่างระหว่างการจำลองกับโลกจริง ข้อมูลที่ได้จะถูกนำมาปรับปรุงผ่านฟังก์ชันรางวัล และเทคนิค Reward Shaping เพื่อเร่งกระบวนการเรียนรู้ โดยผลการศึกษาชี้ให้เห็นว่าระบบ DRL สามารถจัดการกับงานที่ซับซ้อน เช่น การวางแผนเส้นทาง การแซง และการจัดการทางแยกได้อย่างมีประสิทธิภาพ ซึ่งสามารถนำไปประยุกต์ใช้เพื่อยกระดับความสามารถในการขับขี่อัตโนมัติในสถานการณ์จริงได้ โดยจำเป็นต้องพิจารณาแนวทางการเพิ่มความปลอดภัยและประสิทธิภาพในการใช้ข้อมูลตัวอย่างควบคู่กันไป

2.2.9 End-to-End Model-Free Reinforcement Learning for Urban Driving Using Implicit Affordances [25]

งานวิจัยนี้ออกแบบและพัฒนาระบบควบคุมยานยนต์ไร้คนขับแบบ End-to-End สำหรับการขับขี่ในเขตเมืองที่ซับซ้อนโดยใช้เทคนิค DRL ผ่านอัลกอริทึม Rainbow-IQN Ape-X เพื่อเรียนรู้โยบายการขับขี่จากข้อมูลภาพโดยตรงแทนการใช้กฎที่สร้างโดยมนุษย์, โดยนำเสนอเทคนิคใหม่ที่เรียกว่า "Implicit Affordances" เพื่อแก้ปัญหาข้อจำกัดด้านหน่วยความจำและความซับซ้อนของการฝึกฝนโครงข่ายขนาดใหญ่ในระบบ RL ด้วยการแยกกระบวนการออกเป็นสองขั้นตอน คือการใช้ Supervised Learning ฝึกโมเดล ResNet-18 ให้ทำนายข้อมูลเชิงความหมายระดับสูง เช่น การแบ่งส่วนภาพ และสถานะไฟจราจร ก่อนนำฟีเจอร์ที่สกัดได้ไปใช้เป็นอินพุตให้กับระบบ RL แทนการใช้ภาพดิบ ส่วนการควบคุมยานพาหนะใช้การกำหนดฟังก์ชันรางวัล ที่ละเอียดโดยพิจารณาจากความเร็วที่ต้องการตามสถานการณ์ ตำแหน่งกึ่งกลางเลน และมุมเลี้ยวเพื่อควบคุมการกระทำแบบไม่ต่อเนื่องได้อย่างแม่นยำ ผลการทดลองในโปรแกรมจำลอง CARLA แสดงให้เห็นว่าระบบสามารถจัดการกับสถานการณ์ที่ยากลำบาก เช่น การหยุดรอไฟแดง การจัดการทางแยก และการหลบหลีกคนเดินเท้าได้อย่างมีประสิทธิภาพ โดยสามารถชนะเลิศการแข่งขัน CARLA Autonomous Driving Challenge ในหมวด Camera Only และมีประสิทธิภาพเหนือกว่าวิธีการ Imitation Learning แบบดั้งเดิม ซึ่งแสดงถึงศักยภาพในการประยุกต์ใช้ DRL กับงานขับขี่อัตโนมัติที่ต้องอาศัยการประมวลผลการมองเห็นเป็นหลัก

2.2.10 Decision-making Strategy on Highway for Autonomous Vehicles using Deep Reinforcement Learning [26]

งานวิจัยนี้ออกแบบและพัฒนากลยุทธ์การตัดสินใจสำหรับยานยนต์อัตโนมัติบนทางหลวงโดยใช้เทคโนโลยี DRL ผ่านอัลกอริทึม Dueling Deep Q-Network (DDQN) สำหรับจัดการพฤติกรรมการแข่งขันและการเปลี่ยนช่องจราจรที่ซับซ้อน โดยใช้โครงสร้างการควบคุมแบบลำดับขั้น ที่แยกการตัดสินใจเชิงกลยุทธ์ในระดับบนจากการควบคุมความเร็วและความเร่งในระดับล่าง แม้ว่าอัลกอริทึม Deep Q-Network (DQN) แบบดั้งเดิมจะสามารถเรียนรู้ได้แต่ DDQN ได้รับการปรับปรุงโครงสร้างเครือข่ายประสาทเทียมให้แยกการประเมินค่า State-value และ Advantage function ออกจากกันเพื่อเพิ่มความแม่นยำในการเลือกการกระทำ ส่วนการฝึกฝนระบบใช้สภาพแวดล้อมจำลองถนน 3 เลนผ่านชุดเครื่องมือ OpenAI Gym โดยป้อนข้อมูลสถานะความเร็วสัมพัทธ์และระยะห่างระหว่างยานพาหนะเข้าสู่ระบบเพื่อเรียนรู้ผ่านการลองผิดลองถูก ข้อมูลผลลัพธ์จะถูกประเมินผ่านฟังก์ชันรางวัลที่เน้นการหลีกเลี่ยงการชนและประสิทธิภาพความเร็ว โดยผลการทดลองแสดงให้เห็นว่าระบบ DDQN มีอัตราการลู่เข้าสู่ค่าที่เหมาะสมรวดเร็วกว่าและให้ผลตอบแทนสะสมสูงกว่าวิธีการ DQN และโมเดลกฎพื้นฐาน อย่างมีนัยสำคัญ ซึ่งสามารถนำไปประยุกต์ใช้เพื่อยกระดับความปลอดภัยและประสิทธิภาพการตัดสินใจของรถยนต์ไร้คนขับในสถานการณ์จริงที่มีความไม่แน่นอนสูงได้อย่างมีประสิทธิภาพ

2.2.11 Centralized Cooperation for Connected and Automated Vehicles at Intersections by Proximal Policy Optimization [27]

งานวิจัยนี้ออกแบบและพัฒนากระบวนการประสานงานแบบรวมศูนย์สำหรับยานยนต์เชื่อมต่อและขับเคลื่อนอัตโนมัติ บริเวณทางแยกที่ปราศจากสัญญาณไฟจราจรโดยใช้เทคนิค RL ผ่านอัลกอริทึมที่พัฒนาขึ้นใหม่เรียกว่า Model Accelerated Proximal Policy Optimization (MA-PPO) เพื่อแก้ไขปัญหาการคำนวณที่สูงของวิธีการดั้งเดิมอย่าง Model Predictive Control, โดยการบูรณาการแบบจำลองสภาพแวดล้อมล่วงหน้า เข้ากับอัลกอริทึม PPO เพื่อเร่งกระบวนการเรียนรู้และเพิ่มประสิทธิภาพในการใช้ข้อมูลตัวอย่าง การทำงานของระบบครอบคลุมการกำหนดสถานะและการกระทำของยานพาหนะ 8 คันในสถานการณ์จำลองสี่แยกที่มีความขัดแย้งของการจราจรทั้งแบบตัดผ่าน รวมเลน และแยกเลน เพื่อฝึกฝนให้นโยบายการขับที่สามารถหลีกเลี่ยงการชนและผ่านทางแยกได้อย่างรวดเร็ว, โดยผลการทดลองเปรียบเทียบกับวิธี Vehicle-Intersection Coordination Scheme (VICS) ที่ใช้พื้นฐาน MPC แสดงให้เห็นว่าระบบที่นำเสนอใช้เวลาในการคำนวณน้อยกว่าถึง 400 เท่า และช่วยเพิ่มประสิทธิภาพการไหลเวียนของจราจรได้ดีกว่าถึง 4.5 เท่า, นอกจากนี้ยังพบว่า MA-PPO มีอัตราการลู่เข้าสู่ผลลัพธ์ที่เหมาะสมรวดเร็วกว่า PPO แบบปกติถึงสองเท่า ซึ่งแสดงถึงศักยภาพในการนำไปประยุกต์ใช้เพื่อการตัดสินใจควบคุมยานพาหนะแบบเรียลไทม์ที่ต้องการความรวดเร็วและปลอดภัยสูง

2.2.12 Godot Reinforcement Learning Agents [28]

งานวิจัยนี้ออกแบบและพัฒนาอินเทอร์เฟซแบบโอเพนซอร์สสำหรับฝึกฝนตัวแทนปัญญาประดิษฐ์ในสภาพแวดล้อม 2 มิติและ 3 มิติโดยใช้เทคนิค DRL ผ่านเครื่องมือ Godot Game Engine สำหรับสร้างพฤติกรรมการเรียนรู้ที่ซับซ้อนโดยรองรับการทำงานร่วมกับเฟรมเวิร์กมาตรฐานอย่าง Ray Rllib และ Stable Baselines ซึ่งช่วยให้นักวิจัยสามารถเข้าถึงอัลกอริทึมที่มีประสิทธิภาพกว่า 20 ชนิด เช่น PPO, DQN และ SAC ได้ทันที แม้ว่าเฟรมเวิร์กอย่าง Unity ML Agents จะมีความสามารถสูงแต่มีข้อจำกัดด้านลิขสิทธิ์และความยืดหยุ่นในการปรับแต่งโปรโตคอล ส่วนการรับรู้สภาพแวดล้อมของตัวแทนใช้เซนเซอร์แบบ Raycast เพื่อจำลองการทำงานของ LIDAR ในการวัดระยะความลึกแทนการใช้การประมวลผลภาพเต็มรูปแบบที่สิ้นเปลืองทรัพยากร ข้อมูลการสังเกตการณ์จะถูกส่งผ่านการเชื่อมต่อแบบ TCP Socket ไปยังโค้ด Python เพื่อประมวลผลและฝึกฝนโมเดลโดยรองรับการทำงานแบบขนานเพื่อเพิ่มความเร็วในการเรียนรู้ โดยผลการทดสอบแสดงให้เห็นว่าระบบสามารถสร้างปฏิสัมพันธ์ได้สูงถึง 12,000 ครั้งต่อวินาทีบนฮาร์ดแวร์ทั่วไป ซึ่งแสดงถึงประสิทธิภาพในการนำไปประยุกต์ใช้พัฒนาพฤติกรรมตัวละครในเกมหรือการจำลองหุ่นยนต์เคลื่อนที่เพื่อทดสอบอัลกอริทึมก่อนนำไปใช้ในโลกรจริงได้อย่างยืดหยุ่นและมีประสิทธิภาพ

2.2.13 Autonomous Driving System Using CNN and PPO [29]

งานวิจัยนี้ออกแบบและนำเสนอการใช้งานระบบขับเคลื่อนอัตโนมัติแบบผสมผสานโดยประยุกต์ใช้โมเดลการถดถอยด้วย CNN เพื่อทำนายมุมพวงมาลัยจากภาพมุมมองการขับขี่ที่สร้างขึ้นจากโปรแกรมจำลองซึ่งมาพร้อมกับข้อมูลมุมเลี้ยวในหน่วยเรเดียน โดยมีการเปรียบเทียบและบูรณาการการทำงานระหว่างโมเดล Nvidia ที่มีความเหมาะสมกับสภาพถนนที่มีรายละเอียดน้อย และโมเดล MobileNet-V2 ที่มีประสิทธิภาพในสภาพแวดล้อมที่มีรายละเอียดซับซ้อน เพื่อพัฒนาเป็นโมเดล "Ensemble-M" ที่ผสานจุดเด่นของทั้งสองสถาปัตยกรรมเข้าด้วยกัน ข้อมูลภาพจะผ่านกระบวนการเตรียมและเพิ่มขยายข้อมูล (Image Pre-processing and Augmentation) ก่อนเข้าสู่กระบวนการเรียนรู้เพื่อลดความเสี่ยงในการขับขี่ที่ไม่ปลอดภัย ผลการทดลองเปรียบเทียบแสดงให้เห็นว่าเทคนิค Ensemble-M สามารถปรับปรุงความแม่นยำโดยให้ค่าความคลาดเคลื่อนกำลังสองเฉลี่ยต่ำที่สุดเพียง 0.0201 และใช้ทรัพยากรเวลาในการประมวลผลอย่างมีประสิทธิภาพ ซึ่งพิสูจน์ได้ว่าเป็นแนวทางที่ช่วยยกระดับความปลอดภัยและเป็นทางเลือกที่ดีกว่าสำหรับระบบขับเคลื่อนอัตโนมัติเมื่อเทียบกับการใช้โมเดลเดี่ยว

2.2.14 Unity: A General Platform for Intelligent Agents [20]

งานวิจัยนี้เสนอและวิเคราะห์โครงสร้างของแพลตฟอร์มจำลองสถานการณ์ทั่วไป สำหรับการวิจัยปัญญาประดิษฐ์โดยใช้ Unity Engine ร่วมกับชุดเครื่องมือ Unity ML-Agents Toolkit เพื่อสร้างสภาพแวดล้อมการเรียนรู้ที่มีความซับซ้อนสูงทั้งด้านภาพ กายภาพ และสังคม ซึ่งแตกต่างจากแพลตฟอร์มเฉพาะทางเดิมอย่าง Arcade Learning Environment หรือ Project Malmö ที่มีข้อจำกัดด้านความยืดหยุ่นและการปรับแต่งแบบ Black-box, โดยระบบรองรับการฝึกฝน Agents ด้วยอัลกอริทึม DRL ประสิทธิภาพสูง ได้แก่ PPO และ Soft Actor-Critic (SAC) [30] ผ่านการเชื่อมต่อกับ Python API ด้วยโปรโตคอลสื่อสาร gRPC เพื่อรับส่งข้อมูลจากเซนเซอร์รูปแบบต่างๆ ทั้งข้อมูลภาพ และข้อมูลการวัดระยะ แม้ว่าการจำลองใน Unity อาจใช้ทรัพยากรหน่วยความจำมากกว่าสภาพแวดล้อมแบบง่ายอย่าง Atari แต่มีความโดดเด่นในด้าน Sensory Complexity ที่รองรับการเรนเดอร์ภาพสมจริงและการจำลองฟิสิกส์ผ่าน PhysX, ผลการทดสอบมาตรฐาน ในสภาพแวดล้อมตัวอย่างที่หลากหลาย เช่น 3DBall, Crawler และ Obstacle Tower แสดงให้เห็นว่าระบบสามารถเร่งความเร็วการจำลองได้สูงกว่าเวลาจริงถึง 100 เท่า และสามารถฝึกฝนตัวแทนให้มีประสิทธิภาพเหนือกว่าระดับมนุษย์ในงานที่ซับซ้อน ซึ่งแนวทางนี้สามารถนำไปประยุกต์ใช้เพื่อพัฒนาระบบ Artificial General Intelligence และแก้ปัญหาการถ่ายโอนการเรียนรู้จากระบบจำลองสู่หุ่นยนต์จริง ในสภาพแวดล้อมที่มีพลวัตสูงได้อย่างมีประสิทธิภาพ

ตารางที่ 2.1 สรุปงานวิจัยที่เกี่ยวข้องกับแนวทางที่นำเสนอ

Author	Year	Proposed Schemes
S. C. K. Tekouabou et al. [1]	2022	นำเสนอระบบทำนายความพร้อมใช้งานของที่จอดรถโดยใช้โมเดล Ensemble Learning แบบ Bagging Regression ร่วมกับ IoT ซึ่งให้ความแม่นยำสูงกว่าวิธีดั้งเดิม
G. Ali et al. [2]	2020	พัฒนาระบบจอดรถอัจฉริยะโดยใช้ Deep LSTM Network จัดการข้อมูลอนุกรมเวลาเพื่อทำนายจำนวนที่จอดรถว่างผ่านเครือข่าย LoRaWAN และ MQTT
K. Korosec [3]	2019	ออกแบบและทดสอบระบบ Automated Valet Parking (Level 4) โดยบูรณาการเซนเซอร์บนตัวรถเข้ากับโครงสร้างพื้นฐานอัจฉริยะในอาคารจอดรถ

ตารางที่ 2.1 สรุปงานวิจัยที่เกี่ยวข้องกับแนวทางที่นำเสนอ (ต่อ)

Author	Year	Proposed Schemes
W. Kim and I. Jung [4]	2022	เสนอระบบลานจอดรถอัจฉริยะโดยใช้ Edge Cluster Computing และแผนที่แบบตาราง (Grid Map) เพื่อแก้ปัญหาการสื่อสารในพื้นที่อับสัญญาณสำหรับรถยนต์ไร้คนขับ
A. Uzzaman et al. [5]	2025	วิเคราะห์ความปลอดภัยของยานยนต์อัตโนมัติ โดยเน้นการใช้ Sensor Fusion และเปรียบเทียบประสิทธิภาพระหว่างโมเดล YOLO และ CNN ในการตรวจจับวัตถุ
Y. Deng et al. [6]	2021	สำรวจความเสี่ยงและการป้องกันการโจมตีทางไซเบอร์ (Attacks and Defenses) บนระบบ Deep Learning ของรถยนต์ไร้คนขับ ทั้งการโจมตีทางกายภาพและเชิงประพักษ์
L. Liu et al. [7]	2021	วิเคราะห์สถาปัตยกรรมคอมพิวเตอร์สำหรับรถยนต์ไร้คนขับ โดยเปรียบเทียบการใช้ CNN, R-CNN, YOLO และ LSTM บนฮาร์ดแวร์ NVIDIA Drive และ FPGA
B. R. Kiran et al. [8]	2022	สำรวจการประยุกต์ใช้ Deep Reinforcement Learning (DRL) เช่น DQN และ Actor-Critic เพื่อเรียนรู้นโยบายการขับขี่ในสภาพแวดล้อมที่มีความซับซ้อน
M. Toromanoff et al. [25]	2020	พัฒนาระบบขับขี่แบบ End-to-End ด้วยเทคนิค DRL และ Implicit Affordances เพื่อแยกการเรียนรู้ฟีเจอร์ระดับสูง (เช่น ไฟจราจร) ก่อนส่งเข้าสู่กระบวนการตัดสินใจ

ตารางที่ 2.1 สรุปงานวิจัยที่เกี่ยวข้องกับแนวทางที่นำเสนอ (ต่อ)

Author	Year	Proposed Schemes
J. Liao et al. [26]	2020	เสนออัลกอริทึม Dueling Deep Q-Network (DDQN) ร่วมกับการควบคุมแบบลำดับขั้น (Hierarchical Control) สำหรับการตัดสินใจเปลี่ยนเลนบนทางหลวง
Y. Guan et al. [27]	2020	พัฒนาอัลกอริทึม Model Accelerated PPO (MA-PPO) ที่ใช้แบบจำลองสภาพแวดล้อมล่วงหน้าเพื่อเพิ่มความเร็วในการเรียนรู้สำหรับการประสานงานบริเวณทางแยก
E. Beeching et al. [28]	2021	ออกแบบอินเทอร์เฟซสำหรับฝึกฝน DRL Agents บน Godot Game Engine โดยใช้เซนเซอร์ Ray-cast เพื่อเพิ่มความเร็วในการจำลองและลดทรัพยากรการคำนวณ
A. Singh et al. [29]	2025	นำเสนอโมเดล Ensemble-M ที่ผสมผสาน CNN และ MobileNet-V2 เพื่อทำนายมุมพวงมาลัยจากภาพ ซึ่งช่วยลดค่าความคลาดเคลื่อน (MSE) ได้อย่างมีประสิทธิภาพ
A. Juliani et al. [20]	2018	นำเสนอ Unity ML-Agents Toolkit เป็นแพลตฟอร์มสำหรับฝึกฝน AI ที่มีความซับซ้อนทางกายภาพและประสาทสัมผัสสูง รองรับอัลกอริทึม PPO และ SAC
Unity Technologies [21]	-	เอกสารคู่มืออธิบายการตั้งค่าไฟล์ Configuration สำหรับการฝึกฝน Agents ใน Unity ML-Agents Toolkit

บทที่ 3 วิธีการวิจัย

งานวิจัยนี้มุ่งเน้นการศึกษาและพัฒนาระบบจอตลอดอัตโนมัติโดยประยุกต์ใช้เทคนิคการเรียนรู้เสริมกำลังเชิงลึก (Deep Reinforcement Learning: DRL) ด้วยอัลกอริทึม Proximal Policy Optimization (PPO) ภายในสภาพแวดล้อมจำลองที่สร้างขึ้นด้วย Unity Game Engine ซึ่งมีความเหมาะสมในการจำลองทางฟิสิกส์และมีทรัพยากรสนับสนุนที่หลากหลาย โดยมีลำดับขั้นตอนการดำเนินการวิจัยแบ่งออกเป็น 3 ส่วนหลัก ได้แก่ การศึกษาอัลกอริทึมและทฤษฎีที่เกี่ยวข้อง การออกแบบและสร้างสภาพแวดล้อมจำลอง และการทดสอบและวิเคราะห์ผล ดังรายละเอียดต่อไปนี้

3.1 การศึกษาอัลกอริทึมและทฤษฎีที่เกี่ยวข้อง

เพื่อให้ระบบจอตลอดอัตโนมัติสามารถทำงานได้อย่างมีประสิทธิภาพ งานวิจัยนี้ได้เลือกใช้หลักการทางปัญญาประดิษฐ์ขั้นสูง โดยมีรายละเอียดของทฤษฎีและอัลกอริทึมที่นำมาประยุกต์ใช้ ดังนี้

3.1.1 การเรียนรู้เสริมกำลังเชิงลึก (DRL)

เป็นกระบวนการที่ Agent เรียนรู้ที่จะตัดสินใจแบบลำดับ ผ่านการปฏิสัมพันธ์กับสภาพแวดล้อม เพื่อให้ได้รับผลตอบแทนสะสมสูงสุด, อย่างไรก็ตาม การใช้ RL แบบดั้งเดิมที่ใช้ในการเก็บค่าสถานะมักประสบปัญหาเมื่อต้องจัดการกับปริมาณสถานะที่มีมิติสูง เช่น ข้อมูลภาพจากกล้อง งานวิจัยนี้จึงเลือกใช้ การเรียนรู้เสริมกำลังเชิงลึก (DRL) ซึ่งเป็นการบูรณาการโครงข่ายประสาทเทียมเชิงลึก (DNN) เข้ามาทำหน้าที่เป็นตัวประมาณค่าฟังก์ชัน เพื่อแก้ปัญหา Curse of Dimensionality [31], ทำให้ตัวแทนสามารถเรียนรู้นโยบายการขับขี่ จากข้อมูลเซนเซอร์ที่ซับซ้อนได้โดยตรง โดยไม่ต้องอาศัยการกำหนดคุณลักษณะด้วยมนุษย์, กระบวนการนี้สามารถอธิบายได้ด้วยกระบวนการตัดสินใจแบบมาร์คอฟ (Markov Decision Process) ซึ่งประกอบด้วย สถานะ (S), การกระทำ (A), ฟังก์ชันการเปลี่ยนสถานะ (T), และฟังก์ชันรางวัล (R) [8]

3.1.2 Proximal Policy Optimization (PPO)

ในบรรดาอัลกอริทึม DRL ที่มีอยู่ งานวิจัยนี้เลือกใช้ PPO ซึ่งเป็นอัลกอริทึมแบบ Policy Gradient ที่ได้รับความนิยมสูงเนื่องจากมีความเสถียรในการฝึกฝนและง่ายต่อการปรับจูนเมื่อเทียบกับวิธี Trust Region Policy Optimization (TRPO), หลักการสำคัญของ PPO คือการใช้ฟังก์ชันวัตถุประสงค์แบบ Surrogate ที่มีการตัดทอน เพื่อป้องกันการปรับปรุงนโยบาย ที่รุนแรงเกินไปซึ่งอาจทำให้กระบวนการเรียนรู้ล้มเหลว สมการวัตถุประสงค์ของ PPO สามารถเขียนได้ดังนี้

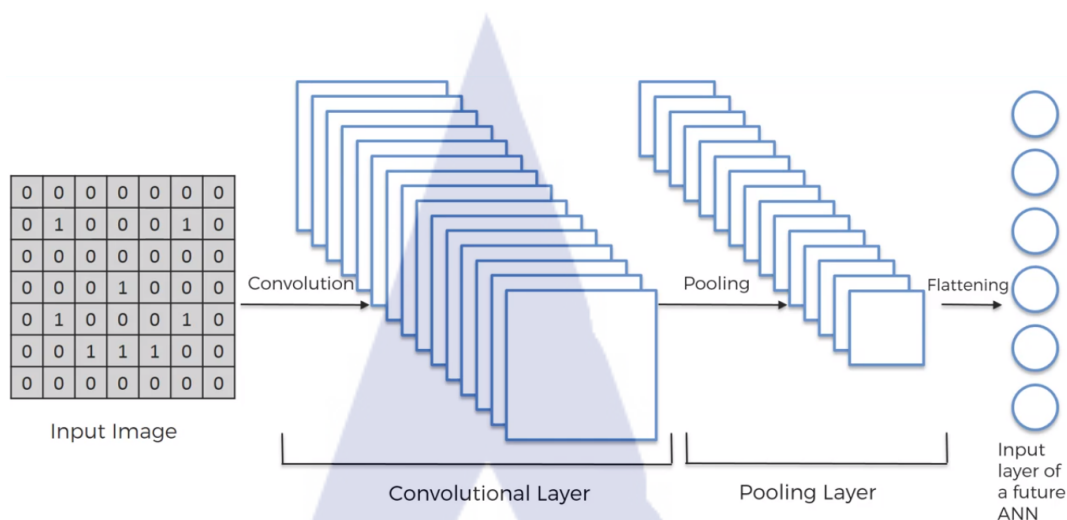
$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (1)$$

โดยที่ $L^{CLIP}(\theta)$ คือ ฟังก์ชันวัตถุประสงค์แบบมีการตัดทอนของ PPO, θ คือ ค่าพารามิเตอร์ปัจจุบันของโครงข่ายนโยบาย, $\hat{E}_t$ คือ ค่าคาดหวังเชิงประจักษ์ (Empirical Expectation) ณ เวลา t , $\min$ คือ ฟังก์ชันคณิตศาสตร์สำหรับเลือกค่าที่ต่ำที่สุดระหว่างสองเงื่อนไข, $r_t(\theta)$ คือ อัตราส่วนความน่าจะเป็นระหว่างนโยบายใหม่กับนโยบายเก่า (Probability Ratio), $\hat{A}_t$ คือ ฟังก์ชันความได้เปรียบประเมิน ณ เวลา t , clip คือ ฟังก์ชันการจำกัดขอบเขตไม่ให้ค่าเกินกว่าช่วงที่กำหนด, ϵ คือ ค่าคงที่ไฮเปอร์พารามิเตอร์ สำหรับกำหนดขอบเขตการเปลี่ยนแปลงนโยบายและ t คือ ลำดับเวลาปัจจุบัน การใช้ PPO ช่วยให้ตัวแทนสามารถเรียนรู้ นโยบายการควบคุมรถต่อเนื่อง ได้อย่างมีประสิทธิภาพและปลอดภัยภายใต้ข้อจำกัดของ สภาพแวดล้อม [25], [26]

3.1.3 โครงข่ายประสาทแบบคอนโวลูชันและการแปลงข้อมูล (CNN and Flattening)

เนื่องจากข้อมูลนำเข้าหลักของระบบเป็นภาพจากกล้อง งานวิจัยนี้จึงประยุกต์ใช้ Convolutional Neural Network (CNN) ซึ่งมีประสิทธิภาพสูงในการสกัดคุณลักษณะ จากข้อมูลภาพ [14], กระบวนการประมวลผลข้อมูลประกอบด้วยขั้นตอนดังนี้:

1. การสกัดคุณลักษณะ: ภาพจากกล้องจะถูกส่งผ่านชั้น Convolutional เพื่อกรองและสกัดรูปแบบที่สำคัญ เช่น ขอบถนน หรือสิ่งกีดขวาง
2. การแปลงข้อมูลให้เป็นเวกเตอร์: ข้อมูลผลลัพธ์จากชั้น CNN ซึ่งอยู่ในรูปแบบเมทริกซ์หลายมิติ จะต้องถูกผ่านกระบวนการ Flatten เพื่อแปลงให้เป็นเวกเตอร์ 1 มิติ กระบวนการนี้มีความสำคัญอย่างยิ่งในการเชื่อมต่อระหว่างส่วนการมองเห็น กับส่วนการตัดสินใจ เพื่อให้สามารถนำข้อมูลภาพไปรวมกับข้อมูลเชิงฟิสิกส์อื่นๆ ก่อนส่งเข้าสู่โครงข่าย Fully Connected Layer ของ อัลกอริทึม PPO ต่อไป [6], [7]



รูปที่ 3.1 Convolutional Layer และ การทำ Flattening

3.2 การออกแบบและสร้างสภาพแวดล้อมจำลอง

การฝึกฝนระบบขับเคลื่อนอัตโนมัติในโลกจริงมีความเสี่ยงสูงและใช้ต้นทุนมาก งานวิจัยนี้จึงใช้ Unity Game Engine เป็นแพลตฟอร์มหลักในการสร้างสภาพแวดล้อมจำลอง โดยมีรายละเอียดการออกแบบดังนี้

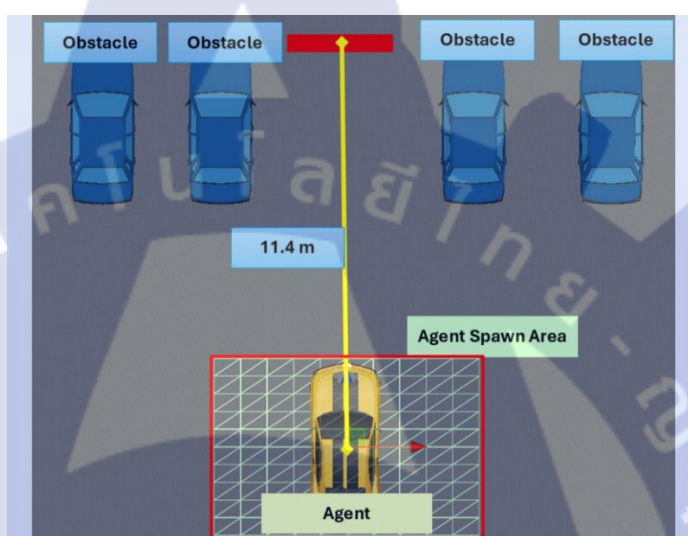
3.2.1 การจำลองทางฟิสิกส์และเครื่องมือ

การฝึกฝนระบบขับเคลื่อนอัตโนมัติในโลกจริงมีความเสี่ยงสูงและใช้ต้นทุนมาก ดังนั้นการใช้แพลตฟอร์มจำลองสถานการณ์ทั่วไป เช่น Unity Game Engine จึงเป็นทางเลือกที่เหมาะสม, Unity มีความโดดเด่นในด้านความซับซ้อนทางกายภาพ ผ่านระบบ Nvidia PhysX ที่สามารถจำลองพลวัตของยานยนต์ การชน และแรงเสียดทานได้อย่างสมจริงนอกจากนี้ การใช้ Unity ร่วมกับ Unity ML-Agents Toolkit ยังช่วยให้สามารถสร้างสภาพแวดล้อมการเรียนรู้ที่ยืดหยุ่น โดยสามารถควบคุมความเร็วของการจำลองให้เร็วกว่าเวลาจริง ซึ่งช่วยเร่งกระบวนการเก็บข้อมูลตัวอย่าง สำหรับฝึกฝนโมเดล DRL ให้รวดเร็วยิ่งขึ้น, รวมถึงรองรับการกำหนดค่าพารามิเตอร์แบบสุ่ม เพื่อเพิ่มความทนทานของโมเดลต่อความไม่แน่นอนเมื่อนำไปใช้งานจริง

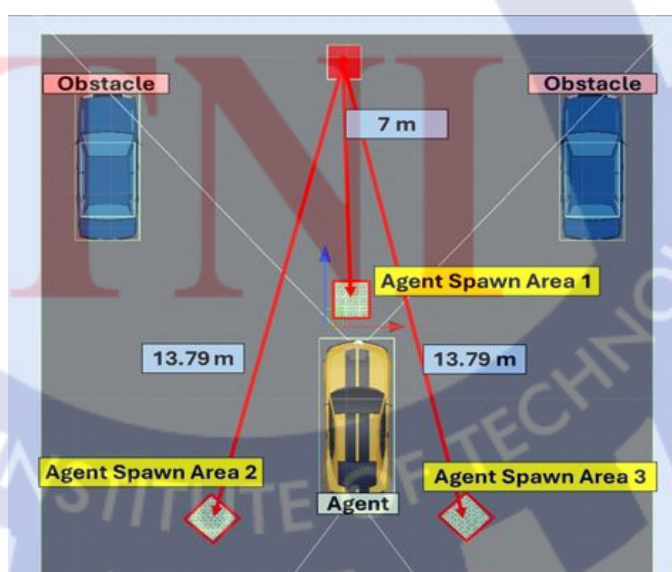
3.2.2 องค์ประกอบของฉากจำลอง

งานวิจัยนี้ใช้ Unity Game Engine ในการสร้างสภาพแวดล้อมจำลอง โดยออกแบบเป็นพื้นที่สี่เหลี่ยมที่มีผนังสีขาวล้อมรอบ องค์ประกอบภายในฉากจำลองประกอบด้วย รถสีเหลืองทำหน้าที่เป็น Agent ที่ต้องเรียนรู้การจอด, รถสีน้ำเงินทำหน้าที่เป็นสิ่งกีดขวาง และกล่องสีแดงเป็นพื้นที่เป้าหมายซึ่งเป็นองค์ประกอบพื้นฐาน ส่วนสิ่งที่แตกต่างคือการกำหนดจุดเริ่มต้นโดยจะมี 2 สภาพแวดล้อม ได้แก่ โดยสภาพแวดล้อมแรก คือจุดเริ่มต้นแบบหันหน้าเข้าหาเป้าหมายอย่างเดียวโดย

จะ Agent จะสุ่มจุด spawn ภายในช่องสี่เหลี่ยมเท่านั้น และสภาพแวดล้อมที่ 2 จุดเริ่มต้นมีทั้งหมดที่ 3 จุดคือแบบหันหน้าเข้าหาเป้าหมายอย่างเดียวและแบบเอียงทำมุม 45 องศาทั้งสองด้าน โดยภายในกรอบการเกิดในจุด Spawn ก็มีการสุ่มภายในกรอบนั้นทำให้ค่า X และ Z เปลี่ยนแปลงตลอดเวลาและมีการสุ่มเพิ่มมุมจาก 45 องศา บวกลบ 20 องศา เพื่อไม่ให้ AI เรียนรู้สถานการณ์ที่แตกต่างไปตลอดเวลาเพื่อเหมาะกับการใช้ การเรียนรู้เสริมกำลังเชิงลึก เพื่อให้ตัวแทนเรียนรู้การเลี้ยวและการจัดระเบียบรถ ดังแสดงในรูปที่ 3.2 และ รูปที่ 3.3



รูปที่ 3.2 ภาพจำลองสภาพแวดล้อมแบบมีจุดเริ่มต้น 1 จุดในมุมมองด้านบน



รูปที่ 3.3 ภาพจำลองสภาพแวดล้อมแบบมีจุดเริ่มต้น 3 จุดในมุมมองด้านบน

3.2.3 การตั้งค่าการจำลอง (Simulation Settings)

เพื่อให้ระบบสามารถเรียนรู้ได้อย่างมีประสิทธิภาพภายใต้ทรัพยากรที่จำกัด ผู้วิจัยได้กำหนดคุณสมบัติของสภาพแวดล้อมการจำลองดังตารางที่ 3.1 โดยมีการแปลงภาพ RGB เป็นภาพ Grayscale ขนาด 84×84 พิกเซล เพื่อลดภาระการประมวลผล และตารางที่ 3.2 แสดงรายละเอียดการตั้งค่าของรถยนต์ที่ Agent เป็นผู้ควบคุม

ตารางที่ 3.1 การตั้งค่าคุณสมบัติของสภาพแวดล้อมการจำลอง

Simulation properties	Applied settings
เวลาจำกัดในแต่ละรอบ	5 seconds
ข้อมูลนำเข้าของตัวแทน	$84 \times 84 \times 1$ (Grayscale image)
การกระทำของ Agent	4 (forward, reverse, turn left, turn right)

ตารางที่ 3.2 การตั้งค่ารถยนต์จำลอง

Car properties	Applied settings
Mass	1080 g
Angular Drag	0.05
Traction Control	0.212
Full Torque Over All Wheels	500
Reverse Torque	100
Maximum Steer Angle	45
Max Handbrake Torque	$3.402823e + 38$
Speed Type	KPH
Top Speed	200
Rev Range Boundary	1
Slip limit	0.3
Brake Torque	20,000

3.3 การทดสอบและวิเคราะห์ผล (Testing and Analysis)

ภายหลังจากที่โมเดลได้รับการฝึกฝนจนเสถียรแล้ว ผู้วิจัยได้ดำเนินการทดสอบประสิทธิภาพ โดยแบ่งลำดับขั้นตอนการทดลองอย่างเป็นระบบ เริ่มต้นจากการทดสอบในสถานการณ์พื้นฐานคือการนำรถเข้าจอดในเส้นทางตรงเพียงอย่างเดียวก่อน โดยทำการทดสอบให้ตัวแทนวิ่งเข้าหาเป้าหมายจำนวน 100 รอบการของการทดลอง ครอบคลุมทั้งกรณีที่ใช้ระบบรางวัลแบบเป้าหมาย และระบบรางวัลแบบจุดใจ เพื่อประเมินผลเบื้องต้น จากนั้นจึงเพิ่มความซับซ้อนของสถานการณ์โดยการกำหนดจุดเริ่มต้นของรถเป็น 3 รูปแบบ ได้แก่ จุดเริ่มต้นทางตรงและจุดเริ่มต้นแบบเฉียง 45 องศาทั้งสองด้าน พร้อมทั้งดำเนินการปรับจูนค่าไฮเปอร์พารามิเตอร์ เพิ่มเติมจนกระทั่งโมเดลในทั้งสองระบบรางวัลสามารถปฏิบัติภารกิจได้ประสบความสำเร็จตามเกณฑ์ที่กำหนด จึงนำผลลัพธ์สุดท้ายมาเก็บรวบรวมข้อมูลสถิติที่สำคัญ ได้แก่ อัตราความสำเร็จ จำนวนครั้งที่ชนสิ่งกีดขวางและระยะเวลาที่ใช้ในการเข้าจอด เพื่อนำมาวิเคราะห์เปรียบเทียบประสิทธิภาพระหว่างโมเดลและสรุปผลการวิจัยในลำดับต่อไป

TNI

TAI - NICHII INSTITUTE OF TECHNOLOGY

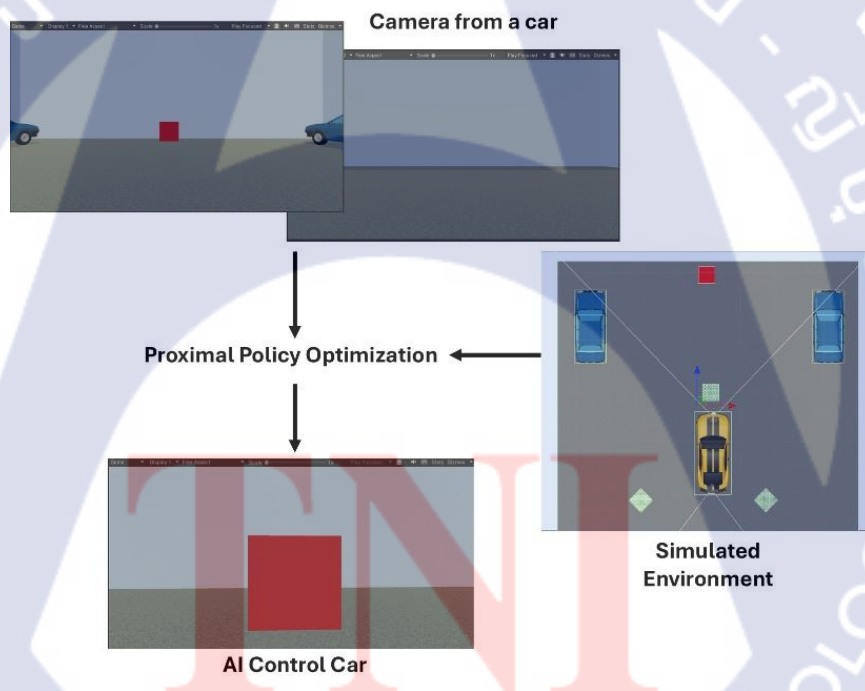
บทที่ 4

ผลการศึกษาและการอภิปรายผล

4.1 ภาพรวมของระบบจอร์รถอัตโนมัติ

4.1.1 แผนผังการทำงาน

ระบบจอร์รถอัตโนมัติเริ่มต้นจากการรับข้อมูลภาพ จากกล้องที่ติดตั้งบริเวณด้านหน้าและด้านหลังของตัวรถ รวมถึงข้อมูลเวกเตอร์สถานะทางกายภาพ ข้อมูลภาพจะถูกแปลงเป็นภาพ Grayscale และปรับขนาดก่อนส่งเข้าสู่กระบวนการประมวลผลหลัก ซึ่งเป็นโมเดลที่ทำงานด้วยอัลกอริทึม Proximal Policy Optimization (PPO) โมเดลจะทำการเรียนรู้และประมวลผลเพื่อส่งคำสั่งควบคุม ไปยังตัวรถ ได้แก่ การเดินทาง ถอยหลัง เลี้ยวซ้าย และเลี้ยวขวา เพื่อนำรถเข้าสู่ช่องจอดเป้าหมาย ดังแสดงกระบวนการทำงานในรูปที่ 4.1



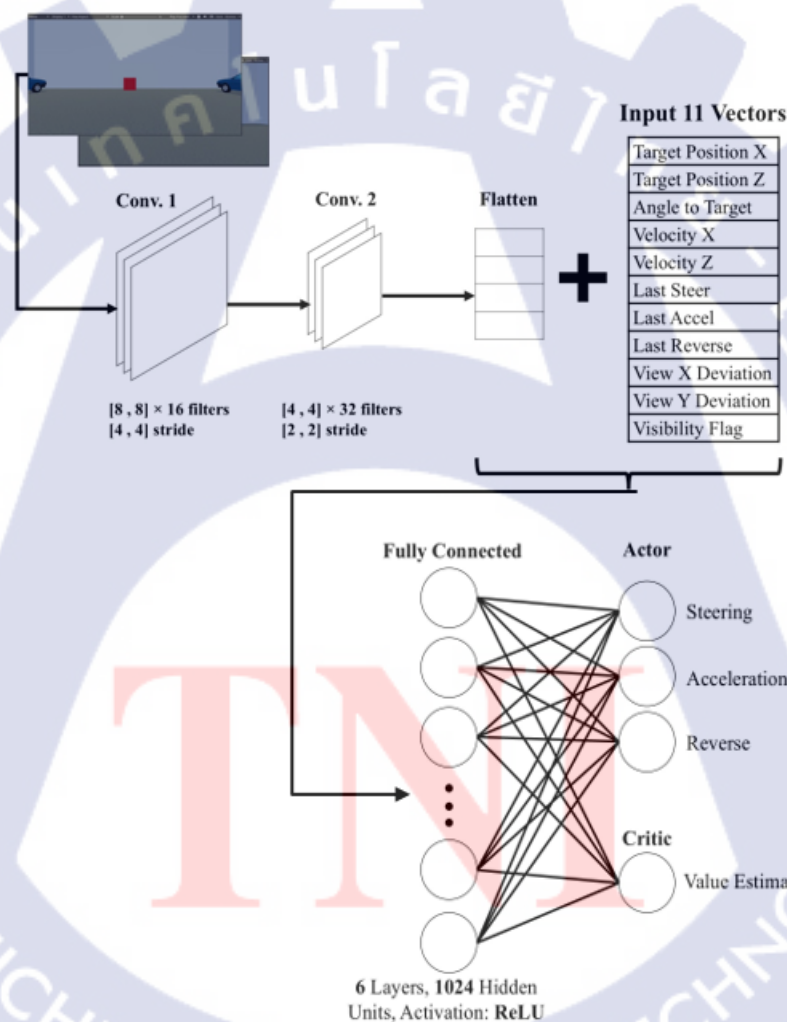
รูปที่ 4.1 แผนผังการทำงานของระบบจอร์รถอัตโนมัติ

4.2 โมเดลอัลกอริทึมและการออกแบบฟังก์ชันรางวัล

4.2.1 สถาปัตยกรรมโครงข่ายประสาทเทียม

งานวิจัยนี้ใช้อัลกอริทึม PPO ภายใต้สถาปัตยกรรมแบบ Actor-Critic ที่รับข้อมูลนำเข้าแบบผสมผสาน ระหว่างข้อมูลภาพและข้อมูลเวกเตอร์ ข้อมูลภาพจะถูกประมวลผลผ่าน CNN เพื่อ

สกัดคุณลักษณะ ในขณะที่ข้อมูลเวกเตอร์จำนวน 11 ค่า ได้แก่ เป้าหมายในแกน X และ Z, มุมของเป้าหมาย, ความเร็วของ Agent ในแกน X และ Z, Action ก่อนหน้าของรถการเลี้ยวซ้าย - ขวา เดินหน้าและถอยหลัง, และค่าของกล้องทั้งแกน X และ Y สุดท้ายจะเป็นค่า Visibility Flag คือค่าไว้สำหรับเช็คได้เห็น Target ในกล้องหรือไม่ ค่าเหล่านี้จะถูกนำมารวมกับคุณลักษณะของภาพเพื่อสร้างตัวแทนสถานะที่สมบูรณ์ ข้อมูลนี้จะถูกส่งเข้าสู่โครงข่าย Fully Connected จำนวน 6 ชั้น Hidden Units ชั้นละ 1024 หน่วย ก่อนจะแยกออกเป็น Actor Head สำหรับกำหนดการกระทำ และ Critic Head สำหรับประเมินค่าความได้เปรียบ ดังแสดงในรูปที่ 4.2



รูปที่ 4.2 แผนผังโมเดล PPO ของตัวแทนและการไหลของข้อมูล

การประมวลผลข้อมูลแบบผสมผสาน นี้มีความสำคัญในการชดเชยข้อจำกัดของเซนเซอร์แต่ละประเภท ข้อมูลเวกเตอร์เป็นข้อมูลเชิงปริมาณที่มีความแม่นยำและมีความเป็นเชิงเส้นสูง ทำให้โมเดลสามารถนำไปคำนวณและเรียนรู้ความสัมพันธ์ด้านพลศาสตร์ ได้อย่างรวดเร็วในขณะที่ข้อมูลภาพมีความซับซ้อนและต้องใช้ทรัพยากรในการคำนวณมากกว่า แต่มีความจำเป็นอย่างยิ่งในการให้บริบทเชิงพื้นที่ เช่น การมองเห็นสิ่งกีดขวางหรือลักษณะของเป้าหมาย ซึ่งข้อมูลเวกเตอร์ไม่สามารถบอกได้ทั้งหมด ส่วนของกระบวนการที่ CNN สามารถรับรู้ได้ว่าบริเวณใดของภาพคือเป้าหมาย นั้น ในระยะเริ่มต้นโมเดลจะยังไม่ทราบถึงความหมายของวัตถุ แต่ชั้น Convolutional จะทำหน้าที่สกัดคุณลักษณะทางภาพพื้นฐาน เช่น เส้นขอบ หรือค่าความเข้มสี เมื่อตัวแทนตัดสินใจเคลื่อนที่เข้าหาพิกเซลลักษณะดังกล่าวแล้วได้รับ Reward กลับมา อัลกอริทึมจะใช้กระบวนการปรับปรุงค่าน้ำหนักย้อนกลับ ส่งผลให้ในการประมวลผลรอบถัดไป CNN จะเรียนรู้ที่จะเน้นความสำคัญ ไปที่กลุ่มพิกเซลที่มีรูปแบบตรงกับกล่องเป้าหมายโดยอัตโนมัติ

4.2.2 การกำหนดค่าไฮเปอร์พารามิเตอร์

ในการฝึกฝนโมเดลสำหรับการจอร์นทั้งแบบตรงและแบบ 45 องศา ผู้วิจัยได้กำหนดค่าไฮเปอร์พารามิเตอร์เพื่อให้การเรียนรู้มีความเสถียรและสอดคล้องกับความซับซ้อนของภารกิจ โดยมี การปรับเปลี่ยนจากค่าเริ่มต้นของชุดเครื่องมือ Unity ML-Agents ในหลายส่วนที่สำคัญ เริ่มจากการกำหนด Learning Rate ไว้ที่ 0.0001 พร้อมรูปแบบ Linear Schedule เพื่อให้การปรับปรุงค่าน้ำหนักค่อย ๆ ลดลงจนเข้าใกล้ศูนย์ ซึ่งช่วยให้โมเดลเข้าสู่จุดสมดุลได้อย่างมีประสิทธิภาพ นอกจากนี้ค่าพารามิเตอร์ Beta ได้ถูกปรับลดลงมาอยู่ที่ 0.001 เพื่อลดการกระทำแบบสุ่ม และกระตุ้นให้ตัวแทนเน้นนโยบายที่เรียนรู้แล้วมาใช้งานจริง มากขึ้น ส่วนค่า Epsilon ตั้งไว้ที่ 0.2 และค่า Lambda ตั้งไว้ที่ 0.95 เพื่อรักษาสมดุลระหว่างความแม่นยำและความเร็วในการอัปเดตนโยบายของอัลกอริทึม PPO สำหรับโครงสร้างของเครือข่ายประสาทเทียมนั้น ผู้วิจัยได้ขยายขนาดขึ้นอย่างมีนัยสำคัญโดยใช้ Hidden Units ถึง 1,024 หน่วยจำนวน 6 ชั้น เพื่อให้เครือข่ายมีขีดความสามารถเพียงพอในการสกัดคุณลักษณะเชิงลึกจากข้อมูลภาพร่วมกับข้อมูลเวกเตอร์ พร้อมกันนี้ได้ขยายขนาด Buffer Size เป็น 40,960 และ Batch Size เป็น 4,096 ควบคู่กับการกำหนด Num Epoch ที่ 15 รอบ เพื่อให้การประมวลผลและเรียนรู้จากประสบการณ์ในแต่ละรอบมีประสิทธิภาพสูงสุด และได้กำหนดค่า Gamma ไว้ที่ 0.999 เพื่อให้ตัวแทนให้ความสำคัญกับผลตอบแทนระยะยาวซึ่งก็คือเป้าหมายการจอร์นให้สำเร็จ

ตารางที่ 4.1 การตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO สำหรับจุดทางตรง

Parameter	Value
trainer_type	ppo
max_steps	2,000,000
learning_rate	0.0001
learning_rate_schedule	linear
beta	0.001
epsilon	0.2
lamdb	0.95
num_epoch	15
buffer_size	40,960
batch_size	4,096
hidden_units	1024
num_layers	6
gamma	0.999

การตั้งค่าทั้งหมดนี้ถูกนำมาใช้งาน ดังแสดงในตารางที่ 4.1 ซึ่งเป็นการตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO ที่สุ่มจุดตั้งโดยทำมุม 0 องศากับเป้าหมาย และในการทดลองอีก 1 รูปแบบคือการสุ่มพื้นที่ตั้งต้นของรถยนต์ 3 พื้นที่ ได้แก่ ทางตรงและเริ่มต้น 45 องศาตามที่แสดงในตารางที่ 4.2 จะเห็นได้ว่าค่า max_steps มีการเพิ่มจาก 2,000,000 steps ไปเป็น 5,000,000 step เพื่อเพิ่ม Episode การเรียนรู้ของ Agent ให้มีระยะเวลาเพียงพอต่อการเรียนรู้ทิศทาง การเลี้ยว และการจัดการกับสภาพแวดล้อมที่ซับซ้อนมากยิ่งขึ้น นอกจากนี้ การกำหนดค่าสูงสุดไว้ที่ 5,000,000 steps ยังเป็นผลมาจากข้อจำกัดทางด้านทรัพยากร โดยเป็นจำนวนสูงสุดที่ฮาร์ดแวร์ซึ่งใช้ในการประมวลผลฝึกฝนโมเดลปัญญาประดิษฐ์ (Train AI) ในงานวิจัยนี้สามารถรองรับการทำงานได้

ตารางที่ 4.2 การตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO สำหรับจุด 45 องศา

Parameter	Value
trainer_type	ppo
max_steps	5,000,000
learning_rate	0.0001
learning_rate_schedule	linear

ตารางที่ 4.2 การตั้งค่าไฮเปอร์พารามิเตอร์ของ PPO สำหรับจุด 45 องศา (ต่อ)

Parameter	Value
beta	0.001
epsilon	0.2
lambda	0.95
num_epoch	15
buffer_size	40,960
batch_size	4,096
hidden_units	1024
num_layers	6
gamma	0.999

4.2.3 การออกแบบระบบรางวัล

เพื่อให้ตัวแทนสามารถเรียนรู้พฤติกรรมที่ซับซ้อนได้ ระบบจำลองได้กำหนดค่ารางวัล เพื่อตอบสนองต่อการกระทำของตัวแทนในสถานะต่างๆ โดยแบ่งเกณฑ์การให้คะแนนพื้นฐานออกเป็น รางวัลเชิงลบเมื่อทำผิดพลาด เช่น การชน การใช้เวลานานเกินไปในการไปถึงเป้าหมาย และรางวัลเชิงบวกเมื่อเข้าใกล้เป้าหมายหรือทำภารกิจสำเร็จ รายละเอียดการกำหนดค่ารางวัลพื้นฐานแสดงดังตารางที่ 4.3

ตารางที่ 4.3 การกำหนดค่ารางวัลพื้นฐานในระบบจำลอง

Agent state	Provide rewards
Time Usage	- 10
Dead	- 2000
Bad Park	- 1000
Goal	+ 2500
Distance reward	- (Distance between the car and the goal.) $\times$ 1.5) - 25
Camera reward	+ 5
Center reward	1 - ((Distance between the goal and the center of the camera.) $\times$ 2)

1. รางวัลเมื่อถึงเป้าหมาย: ให้คะแนนบวกเมื่อจอดสำเร็จและคะแนนลบเมื่อชนหรือจอดผิดทิศทาง
2. รางวัลแบบจูงใจ: เพิ่มรางวัลย่อยในทุกขั้นตอนการกระทำ เช่น การเคลื่อนที่เข้าใกล้เป้าหมาย การหันหน้ารถให้ตรง และการจัดระเบียบมุมรถ ดังรายละเอียดในตารางที่ 4.4

ตารางที่ 4.4 การกำหนดรางวัลแบบจูงใจ

Agent state	Provide rewards
Approach target	(Difference between current distance and distance of the previous event.) $\times$ 2.0
Face right direction	Facing directly towards the target. $\times$ 0.1
Drive right direction	Steering in the direction towards the goal. $\times$ 0.2
Angle Alignment	Function COS (The angle of the vehicle relative to the target.) $\times$ 0.3
Proximity (< 5m)	$2 \times (5 - \text{Current distance of the car.})$
Within Range (< 10m)	+ 0.1
Moving Away	- 0.2

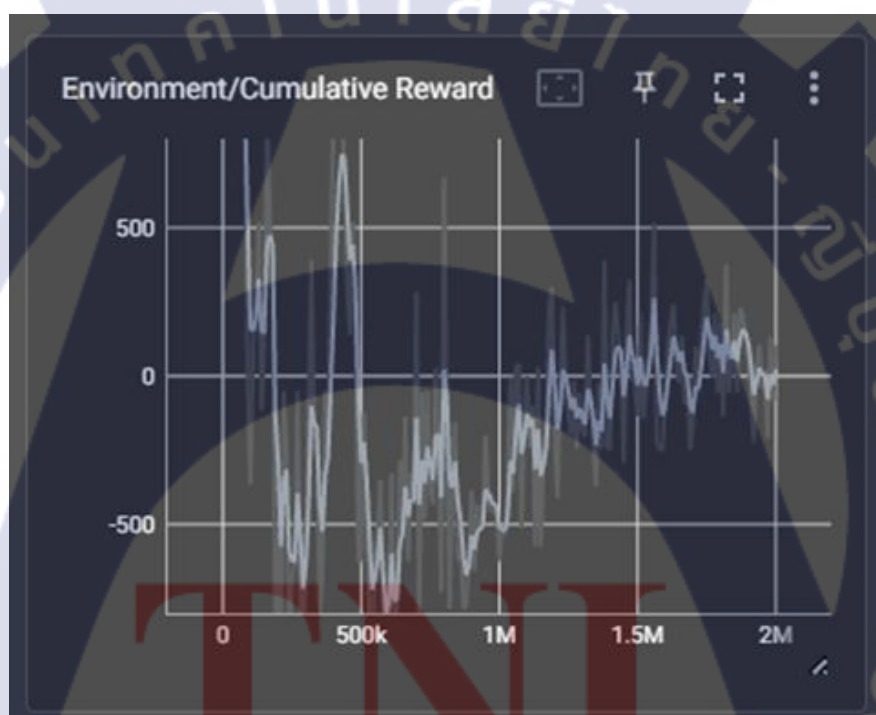
การรับผลตอบแทนภายในรอบการเรียนรู้แต่ละรอบ ซึ่งถูกจำกัดเวลาการทำงานไว้ที่ 5 วินาทีนั้น จะมีรูปแบบการให้คะแนนที่แตกต่างกันตามระบบการออกแบบ ในระบบรางวัลแบบให้ครั้งเดียว Agent จะไม่ได้รับคะแนนใด ๆ ระหว่างการเคลื่อนที่ แต่จะได้รับคะแนนสะสมครั้งเดียว ก็ต่อเมื่อการกระทำสิ้นสุดลง หรือเมื่อหมดเวลา 5 วินาที เช่น ได้รับคะแนน +2500 เมื่อเข้าจอดสำเร็จ หรือถูกหัก -2000 ทันทีเมื่อเกิดการชน กลไกนี้ทำให้การเรียนรู้ในช่วงแรกเป็นไปได้ช้าและยากลำบาก เนื่องจากตัวแทนไม่ทราบว่าการตัดสินใจหักพวงมาลัยหรือเร่งความเร็วในระหว่างทางนั้นถูกต้องหรือไม่

ในทางกลับกัน ระบบรางวัล Reward Shaping จะทำการประเมินพฤติกรรมและให้คะแนนย่อย ทันทีในทุก ๆ สเต็ปของการกระทำ ตลอดระยะเวลา 5 วินาที โดยไม่ต้องรอให้จบ Episode เช่น หากตัวแทนสั่งการหักพวงมาลัยเข้าหาเป้าหมายได้ถูกต้อง หรือระยะห่างจากเป้าหมายลดลง จะได้รับคะแนนบวกสะสมเพิ่มขึ้นแบบเรียลไทม์ การป้อนข้อมูลย้อนกลับอย่างต่อเนื่อง ทันทีหลังการกระทำนี้ ทำหน้าที่เสมือนเข็มทิศนำทางที่ช่วยให้ตัวแทนประเมินความถูกต้องของพฤติกรรมตนเองได้อย่างรวดเร็ว ส่งผลให้ตัวแทนหลีกเลี่ยงการหยุดนิ่งอยู่กับที่จนหมดเวลา และเพิ่มอัตราความสำเร็จในการเข้าจอดได้อย่างมีประสิทธิภาพ

4.3 การวิเคราะห์ความเสถียรของกระบวนการเรียนรู้

4.3.1 ผลของการฝึกฝนแบบจำลองในสถานการณ์จำลองการจอดรถแนวตรงโดยใช้ระบบรางวัลแบบให้ครั้งเดียว

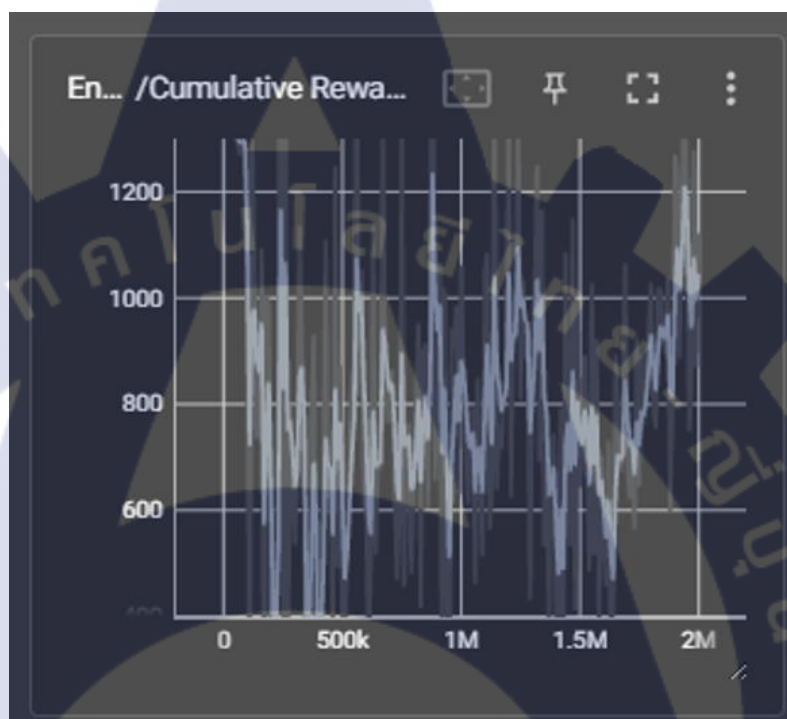
โดยเข้าจอดถูกเงื่อนไขซึ่งได้รับการประเมินผลตลอดระยะเวลา 2 ล้านขั้นตอน ได้แสดงไว้ดังภาพที่ 4.3 การวิเคราะห์ค่า Cumulative Reward ซึ่งทำหน้าที่เป็นดัชนีชี้วัดความสำเร็จหลัก เผยให้เห็นถึงพัฒนาการด้านประสิทธิภาพการเรียนรู้ที่มีนัยสำคัญ โดยในช่วงเริ่มต้นของการฝึกฝน ค่ารางวัลมีความผันผวนสูงและมีค่าเป็นลบ ซึ่งสะท้อนให้เห็นถึงความสามารถในการจอดรถให้สำเร็จ Agent อย่างไม่กี่ตาม ในช่วงท้ายของการฝึกฝน ค่ารางวัลเฉลี่ยมีแนวโน้มเป็นบวกและเริ่มคงที่ ซึ่งแสดงให้เห็นว่าตัวแทนสามารถเรียนรู้และควบคุมเส้นทางการจอดรถที่ถูกต้องได้สำเร็จ



รูปที่ 4.3 ผลการทดลองการจอดรถทางตรงโดยใช้ระบบรางวัลแบบให้ครั้งเดียว

ในทางตรงกันข้าม ภาพที่ 4.4 นำเสนอตัวชี้วัดการฝึกฝนสำหรับสถานการณ์การจอดรถแนวตรงโดยใช้ฟังก์ชันรางวัลแบบมีรางวัลแรงจูงใจ ซึ่งมีการปรับแต่งรางวัลโดยอิงตามระยะห่างจากเป้าหมาย ผลลัพธ์ที่ได้แสดงให้เห็นถึงประสิทธิภาพที่เหนือกว่าและแตกต่างอย่างชัดเจนเมื่อเปรียบเทียบกับวิธีรางวัลแบบให้เมื่อถึงเป้าหมายครั้งเดียว จุดแตกต่างสำคัญปรากฏให้เห็นในค่ารางวัลสะสม ซึ่งยังคงอยู่ในระดับสูงและเป็นบวกอย่างสม่ำเสมอ โดยมีค่าระหว่าง 600 ถึง 1,200 ตั้งแต่

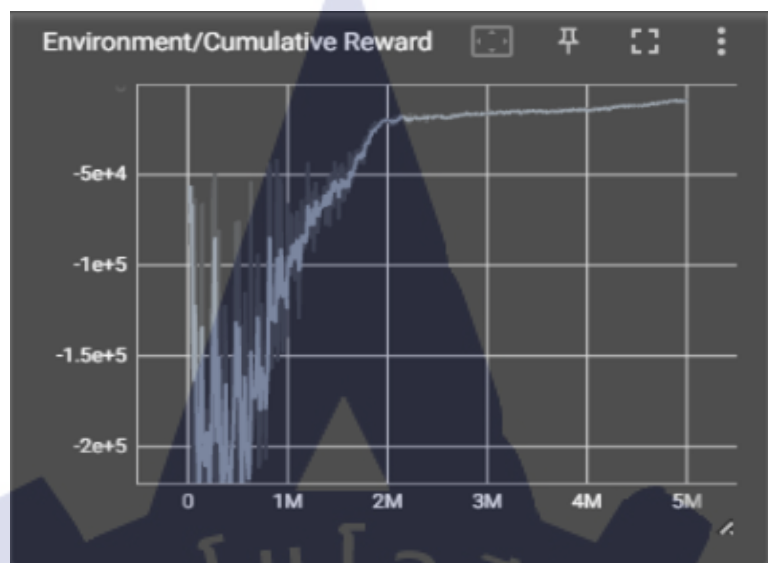
เริ่มต้นการฝึกฝน แตกต่างจากการทดลองก่อนหน้านี้ แบบจำลองไม่จำเป็นต้องใช้ระยะเวลานานในการกู้คืนจากค่ารางวัลที่เป็นลบ ซึ่งบ่งชี้ว่าระบบรางวัลแบบหนาแน่นช่วยให้ตัวแทนสามารถค้นพบนโยบายที่มีประสิทธิภาพได้เกือบจะทันที แม้ว่าความยาวของเหตุการณ์จะสูงในช่วงแรก แต่ก็ได้ลดลงอย่างมีนัยสำคัญและเริ่มคงที่หลังจากผ่านไปประมาณ 500,000 ขั้นตอน แสดงถึงการปรับปรุงกระบวนการจอตลอดให้เหมาะสมอย่างรวดเร็ว



รูปที่ 4.4 ผลการทดลองการจอตลอดทางตรงโดยใช้ระบบรางวัลแบบมีแรงจูงใจ

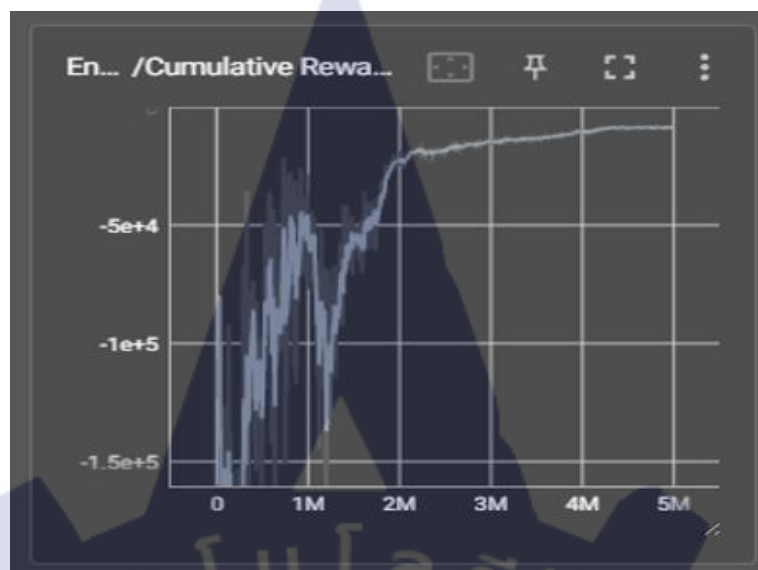
4.3.2 ผลการทดลองการจอตลอดเชิง 45 องศา

ในการทดลองนี้ ฟังก์ชันรางวัลได้รับการกำหนดค่าให้มีความสำคัญกับความสำเร็จที่ปลายทางเป็นหลัก โดย Agent จะได้รับรางวัลเมื่อเสร็จสิ้นภารกิจ เสริมด้วยแรงจูงใจจากการจัดตำแหน่งกล้องอย่างต่อเนื่องเพื่อช่วยในการกำหนดทิศทางของยานพาหนะ แต่ปราศจากความช่วยเหลือจากรางวัลแบบปรับแต่งตามระยะทาง ตัวชี้วัดการฝึกฝนตลอดระยะเวลา 5 ล้านขั้นตอน ดังที่แสดงในภาพที่ 4.5 เผยให้เห็นเส้นโค้งการเรียนรู้ที่ชัดเจน ในช่วงระยะเริ่มต้น (0 ถึง 1.5 ล้านขั้นตอน) ผลลัพธ์ได้เน้นย้ำถึงความท้าทายโดยธรรมชาติของการเรียนรู้โดยปราศจากการชี้นำด้านระยะทาง



รูปที่ 4.5 ผลการทดลองการจอตกรโดยสุ่มจุดเกิด 3 จุดโดยใช้ระบบรางวัลแบบให้ครั้งเดียว

งานวิจัยรูปแบบสุดท้ายเพื่ออำนวยความสะดวกในการเรียนรู้ภายในสภาพแวดล้อมนี้ ฟังก์ชันรางวัลได้รับการปรับปรุงให้ดียิ่งขึ้น โดยนอกเหนือจากรางวัลจากการเข้าจอดสำเร็จเพียงอย่างเดียว ผู้ได้นำเสริมคะแนนแรงจูงใจจากการจัดตำแหน่ง มาใช้เพิ่มเติม กลไกนี้ทำหน้าที่ให้ข้อมูลป้อนกลับอย่างต่อเนื่องโดยอิงจากการจัดตำแหน่งของกล่องตัวแทนเทียบกับจุดกึ่งกลางของช่องจอดเป้าหมาย ซึ่งเป็นการกระตุ้นให้ยานพาหนะปรับทิศทางของตนเองให้ถูกต้องตั้งแต่ระยะเริ่มต้นของการเข้าจอด ผลการฝึกฝนตลอดระยะเวลา 5 ล้านขั้นตอน ดังภาพที่ 4.6 แสดงให้เห็นถึงเส้นโค้งการเรียนรู้ที่แตกต่างจากสถานการณ์การจอตกรแนวตรงอย่างชัดเจน ในช่วงระยะเริ่มต้น (0 ถึง 1.5 ล้านขั้นตอน) ตัวแทนแสดงความผันผวนสูงพร้อมกับค่ารางวัลที่เป็นลบ ซึ่งสะท้อนถึงความยากลำบากในการหามุมเข้าจอดที่ปลอดภัยโดยไม่เกิดการชน อย่างไรก็ตาม แรงจูงใจจากการจัดตำแหน่งได้ให้คำแนะนำที่จำเป็น ซึ่งช่วยป้องกันไม่ให้ตัวแทนเกิดสภาวะติดขัดดังที่อาจเกิดขึ้นในระบบรางวัลแบบบางแนวโน้มขาขึ้นที่ชัดเจนปรากฏขึ้นหลังจาก 2 ล้านขั้นตอน ซึ่งเป็นจุดที่ตัวแทนสามารถ "แก้ไขข้อผิดพลาด" ของสภาพแวดล้อมได้อย่างมีประสิทธิภาพ และสามารถรักษาระดับรางวัลให้สูงและคงที่ได้หลังจากนั้น



รูปที่ 4.6 ผลการทดลองการจอดรถโดยผู้ขับขี่เกิด 3 จุดโดยใช้ระบบรางวัลแบบมีแรงจูงใจ

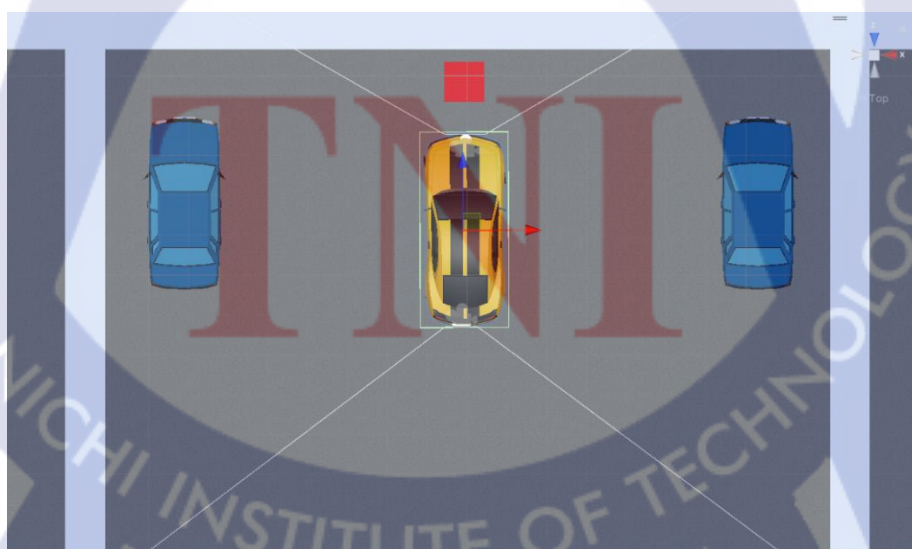
4.4 การทดลอง

ในการดำเนินการทดสอบประสิทธิภาพของระบบจอดรถอัตโนมัติที่พัฒนาขึ้นด้วยอัลกอริทึม Proximal Policy Optimization (PPO) ได้มีการกำหนดทดลองในสภาพแวดล้อมจำลอง ที่ถูกออกแบบมาให้ครอบคลุมความหลากหลายของสถานการณ์เริ่มต้นถึง 3 รูปแบบ ได้แก่ การเริ่มต้นจอดในแนวตรง (0 องศา) และการเริ่มต้นในมุมที่ต้องอาศัยการหักเลี้ยวซับซ้อนอย่างการจอดเอียง 45 องศา (ทั้งด้านซ้ายและด้านขวา) เพื่อทดสอบขีดความสามารถในการวางแผนเส้นทาง ของ Agent ปัญญาประดิษฐ์ การทดลองนี้มุ่งเน้นการเปรียบเทียบเชิงลึกระหว่างสองสถาปัตยกรรมรางวัล คือ โมเดลที่ใช้ระบบรางวัลเป้าหมายเพียงอย่างเดียว ซึ่งให้คะแนนเฉพาะเมื่อทำการกิจสำเร็จ และโมเดลที่ใช้ระบบรางวัลแบบจูงใจต่อเนื่อง ที่มีการให้ผลป้อนกลับตลอดการเคลื่อนที่โดยในแต่ละสถานการณ์จะทำการทดสอบซ้ำจำนวน 100 Episodes เพื่อขจัดความแปรปรวนทางสถิติและเพื่อให้มั่นใจว่าผลลัพธ์ที่ได้เกิดจากการเรียนรู้ที่แท้จริงไม่ใช่ความบังเอิญ ทั้งนี้ เพื่อให้การประเมินผลมีมาตรฐานที่ชัดเจนและสามารถวัดผลในเชิงปริมาณได้ เราได้กำหนดเกณฑ์ความสำเร็จ โดยภารกิจจะถูกรับว่า สำเร็จ ก็ต่อเมื่อยานพาหนะสามารถปฏิบัติตามเงื่อนไขบังคับครบทั้ง 4 ประการ ได้แก่ ด้านความปลอดภัย คือ ยานพาหนะต้องเคลื่อนที่เข้าสู่ช่องจอดโดยปราศจากการเฉี่ยวชนขอบทางหรือสิ่งกีดขวางใดๆ ตลอดทั้งกระบวนการ ต่อไปคือด้านความแม่นยำของตำแหน่ง โดยจุดกึ่งกลางของตัวรถ จะต้องมียระยะห่างจากจุดกึ่งกลางของเป้าหมายไม่เกิน 0.5 เมตร ซึ่งเป็นระยะที่ยอมรับได้ในการใช้งานจริง ส่วนด้านความแม่นยำของทิศทาง แกนของตัวรถจะต้องขนานกับเส้นขอบของช่องจอด โดยอนุญาตให้มีความคลาดเคลื่อนขององศาได้ไม่เกิน 5 องศา เพื่อรับประกันความเรียบร้อยในการจอดและสุดท้าย คือด้าน

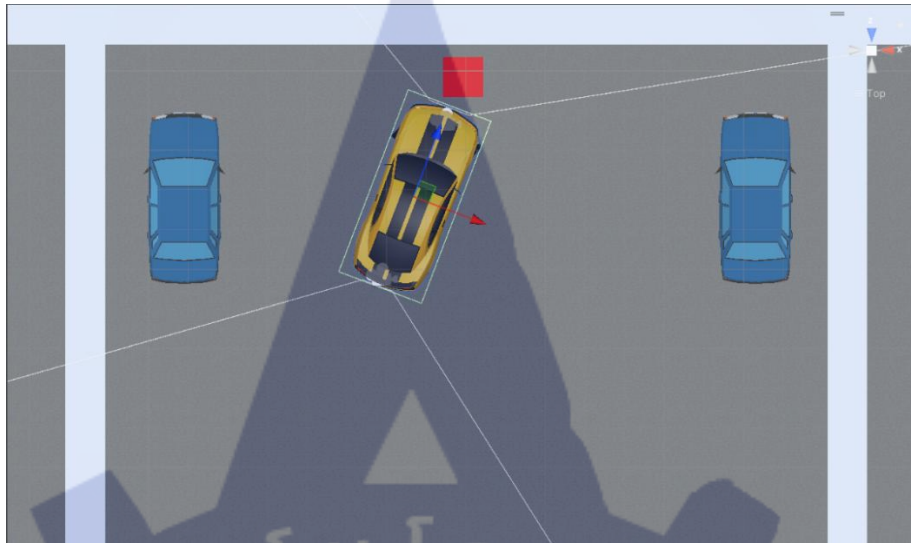
เวลาการดำเนินการทั้งหมดจะต้องเสร็จสิ้นภายในระยะเวลาที่กำหนด 5 วินาที เพื่อแสดงถึงความรวดเร็วในการตัดสินใจ หากยานพาหนะล้มเหลวในการปฏิบัติตามเงื่อนไขข้อใดข้อหนึ่งแม้เพียงข้อเดียว ระบบจะตัดสินใจว่าการทดสอบในรอบนั้นล้มเหลวทันที ซึ่งผลลัพธ์จากการเปรียบเทียบประสิทธิภาพตามเกณฑ์ดังกล่าวได้ถูกรวบรวมและแสดงไว้ในตารางที่ 4.5

ตารางที่ 4.5 ผลการเปรียบเทียบประสิทธิภาพระบบจอร์อัลต์โนมิติในสถานการณ์จอดเข้าช่อง 45 องศา

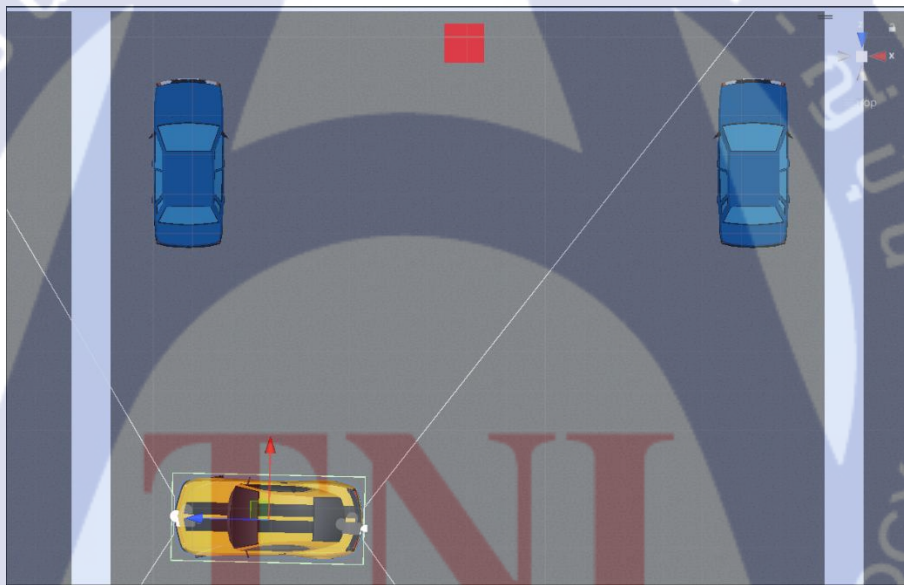
Measurement	Experimental results	
	Only goal	Continuous
Number of Experiments	100	100
Success count	69	98
Success rate	69 %	98 %
Crash count	1	0
Crash rate	1 %	0 %
Time out	30	2
Time out rate	30 %	2 %
Maximum distance	13.79 m	13.79 m



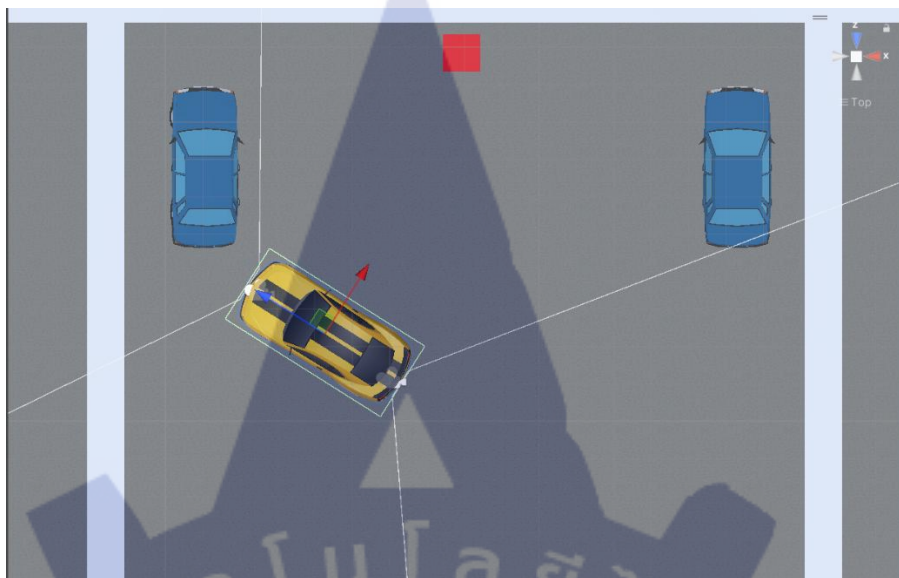
รูปที่ 4.7 ตัวอย่างการเข้าจอดที่สำเร็จทุกเงื่อนไข



รูปที่ 4.8 ตัวอย่างการเข้าจอดที่สำเร็จแต่ไม่ผ่านเงื่อนไขการจอดให้ตรงกับเป้าหมาย



รูปที่ 4.9 ตัวอย่างการเข้าจอดที่ไม่สำเร็จโดยชนกับแพง



รูปที่ 4.10 ตัวอย่างการเข้าจอดที่ไม่สำเร็จโดยชนกับรถยนต์ที่จอดอยู่

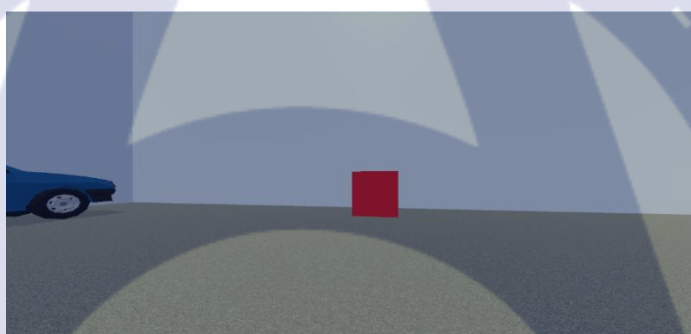
4.5 การวิเคราะห์ผลการทดลอง

จากข้อมูลในตารางที่ 4.6 และรูปที่ 4.7 เมื่อนำผลการทดลองมาเปรียบเทียบประสิทธิภาพระหว่างสองระบบ พบว่าโมเดลที่ใช้ระบบรางวัลแบบจุดใจ มีประสิทธิภาพสูงกว่าโมเดลที่ใช้รางวัลเป้าหมายเพียงอย่างเดียว อย่างมีนัยสำคัญทางสถิติ โดยสามารถทำอัตราความสำเร็จได้สูงถึงร้อยละ 98 ในขณะที่โมเดลแบบดั้งเดิมมีอัตราความสำเร็จเพียงร้อยละ 69 ซึ่งคิดเป็นส่วนต่างความสำเร็จที่ห่างกันถึงร้อยละ 29 ผลลัพธ์นี้มีความสอดคล้องกับการวิเคราะห์ความผิดพลาดที่เกิดขึ้น ซึ่งพบว่าระบบรางวัลเป้าหมายมีอัตราการหมดเวลา สูงถึงร้อยละ 30 อันเป็นเครื่องบ่งชี้ว่า Agent มักติดอยู่ในสภาวะ "Local Optimum" กล่าวคือ ตัวแทนตัดสินใจที่จะหยุดนิ่งหรือไม่กล้าเคลื่อนที่เข้าหาเป้าหมายเพื่อหลีกเลี่ยงการถูกหักคะแนนจากการเดินเวลาหรือเพื่อเลี่ยงความเสี่ยงในการชน ซึ่งเป็นข้อจำกัดเชิงโครงสร้างที่สำคัญของระบบที่ให้รางวัลแบบรวบยอดเพียงครั้งเดียว ในทางตรงกันข้าม ระบบรางวัลแบบจุดใจสามารถแก้ปัญหาได้อย่างมีประสิทธิภาพ โดยลดอัตราการหมดเวลาเหลือเพียงร้อยละ 2 และไม่เกิดการเฉียดชนเลย เนื่องจากระบบมีการป้อนคะแนนย่อยตลอดเวลาเมื่อตัวแทนมีการกระทำที่ถูกต้อง เช่น การเคลื่อนที่เข้าใกล้เป้าหมายหรือการหันหน้ารถได้ตรงทิศทาง ทำให้ตัวแทนสามารถเรียนรู้นโยบายการขับขีที่ถูกต้องได้รวดเร็วและแม่นยำกว่า ยิ่งไปกว่านั้น เมื่อพิจารณาเจาะลึกถึงพฤติกรรมการเคลื่อนที่ เพื่อวิเคราะห์กระบวนการตัดสินใจ พบความแตกต่างเชิงพฤติกรรมที่ชัดเจน โดยโมเดลที่ใช้รางวัลแบบจุดใจแสดงพฤติกรรมการตีวงเลี้ยวที่ราบรื่นและมีการวางแผนปรับแต่งทิศทางหน้า-หลัง เพื่อตั้งลำให้ตัวรถขนานกับช่องจอดตั้งแต่ระยะไกล ส่งผลให้สามารถถอยเข้าจอดได้อย่างแม่นยำและเป็นธรรมชาติ แตกต่างจากโมเดลแบบดั้งเดิมที่มักแสดงพฤติกรรมการเคลื่อนที่ใน

ลักษณะสับสน หรือมีการเลี้ยวแบบหักศอกกะทันหันเกินความจำเป็น ซึ่งก่อให้เกิดปัญหาหม้ออบในการเข้าจอดและต้องใช้เวลาในการจัดระเบียบบรรณานกว่าอย่างเห็นได้ชัด ข้อมูลเชิงประจักษ์เหล่านี้ยืนยันว่าการออกแบบฟังก์ชันรางวัลแบบต่อเนื่องไม่ได้เพียงแค่ช่วยให้ภารกิจสำเร็จเท่านั้น แต่ยังช่วยให้ Agent เข้าใจวิธีการขับซี้และยุทธวิธีในการเข้าจอดที่ถูกต้องและปลอดภัยอย่างแท้จริง ซึ่งเหนือกว่าการมุ่งเน้นเพียงผลลัพธ์สุดท้ายเพียงอย่างเดียว



(a)



(b)



(c)

รูปที่ 4.11 รูปของ Agent ในมุมมองบุคคลที่ 1 (a) Agent เมื่อเริ่ม Episode (b) Agent เคลื่อนที่เข้าใกล้เป้าหมาย (c) Agent เมื่อถึงเป้าหมาย

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบจอตลอดอัตโนมัติโดยประยุกต์ใช้เทคโนโลยีการเรียนรู้แบบเสริมกำลังเชิงลึก (Deep Reinforcement Learning) ด้วยอัลกอริทึม Proximal Policy Optimization (PPO) ภายในสภาพแวดล้อมจำลอง Unity ซึ่งจากการดำเนินการวิจัยและทดสอบระบบ สามารถสรุปผลได้สอดคล้องกับวัตถุประสงค์ที่ตั้งไว้ ดังนี้:

1. ด้านประสิทธิภาพของอัลกอริทึม การใช้ข้อมูลภาพจากกล้องหน้าและกล้องหลังร่วมกับอัลกอริทึม PPO พิสูจน์แล้วว่าเพียงพอต่อการควบคุมรถยนต์ให้เข้าจอดได้โดยไม่ต้องพึ่งพาเซนเซอร์วัดระยะราคาแพง โดย Agent สามารถเรียนรู้ที่จะตีวงเลี้ยวและควบคุมความเร็วได้อย่างเป็นธรรมชาติ คล้ายมนุษย์
2. ด้านการเปรียบเทียบรูปแบบการเรียนรู้ ผลการเปรียบเทียบแสดงให้เห็นว่า การออกแบบฟังก์ชันรางวัล เป็นปัจจัยที่สำคัญต่อความสำเร็จโดยระบบที่ใช้ รางวัลแบบมุ่งใจต่อเนื่อง มีประสิทธิภาพเหนือกว่าระบบรางวัลเมื่อถึงเป้าหมายอย่างเดียวย่างมีนัยสำคัญ โดยมีอัตราความสำเร็จสูงถึงร้อยละ 98 เทียบกับร้อยละ 69 และสามารถลดอัตราการชนเหลือร้อยละ 0 ซึ่งแสดงให้เห็นว่าการป้อนข้อมูลย้อนกลับ อย่างต่อเนื่องช่วยป้องกันปัญหา Local Optimum ได้อย่างมีประสิทธิภาพ

5.2 ปัญหาและข้อจำกัดของการวิจัย

ในการดำเนินงานวิจัยนี้ ผู้วิจัยได้พบข้อจำกัดบางประการที่อาจส่งผลต่อการนำผลลัพธ์ไปประยุกต์ใช้ดังนี้ ข้อแรกคือช่องว่างระหว่างโลกจำลองและโลกจริง แม้ Unity จะจำลองฟิสิกส์ได้ดี แต่ความซับซ้อนของแสงเงา พื้นผิวถนนและแรงเสียดทานในโลกจริงอาจมีความแปรปรวนมากกว่าในโปรแกรมจำลอง ซึ่งอาจทำให้โมเดลที่ฝึกมาดีแล้วเกิดความผิดพลาดเมื่อนำไปใช้จริงทันที ข้อต่อมาคือจำกัดของมุมมองกล้อง การใช้กล้องเพียง 2 ตัวกล้องหน้าและกล้องหลัง อาจเกิดจุดบอด ในบางมุมมองศา โดยเฉพาะขณะเลี้ยวในพื้นที่แคบ ซึ่งต่างจากการใช้เซนเซอร์รอบคัน 360 องศา

5.3 ข้อเสนอแนะ

5.3.1 การพัฒนาอัลกอริทึมและโครงสร้างโมเดล

ควรมีการศึกษาและปรับปรุงโครงสร้างของโครงข่ายประสาทเทียม (Neural Network Structure) ภายในอัลกอริทึม PPO ให้มีความเหมาะสมยิ่งขึ้น เช่น การปรับจำนวนไฮเปอร์พารามิเตอร์ หรือการปรับปรุงระบบการให้คะแนน ให้มีความละเอียดและครอบคลุมเงื่อนไขการขับขี่ที่หลากหลาย

เพื่อเพิ่มความแม่นยำและความเร็วในการเรียนรู้ของตัวแทนให้ดียิ่งขึ้นสำหรับการรองรับสถานการณ์ที่ซับซ้อนกว่าเดิม

5.2.2 การปรับปรุงสภาพแวดล้อมการจำลอง

เพื่อให้ระบบมีความแข็งแกร่งและพร้อมสำหรับการใช้งานจริง ควรมีการพัฒนาสภาพแวดล้อมในโปรแกรมจำลองให้มีความท้าทายมากขึ้น โดยการเพิ่มสิ่งกีดขวางที่มีความยาก หรือจำลองสถานการณ์ที่มีพลวัต เช่น การมีรถคันอื่นเคลื่อนที่ไปมา หรือสภาพแสงและสภาพอากาศที่เปลี่ยนแปลงไป เพื่อทดสอบความสามารถในการปรับตัวและการตัดสินใจของระบบภายใต้ความไม่แน่นอน ซึ่งเป็นปัจจัยสำคัญที่พบได้ทั่วไปในการขับเคลื่อนท้องถนนจริง

5.2.3 การเตรียมความพร้อมสู่การทดสอบในโลกจริง

แม้ผลการทดลองในสภาพแวดล้อมจำลอง จะให้ผลลัพธ์ที่ดีเยี่ยม แต่การนำไปประยุกต์ใช้กับรถยนต์จริง ยังคงมีความท้าทายเรื่องความแตกต่างระหว่างโลกจำลองและโลก ดังนั้น งานวิจัยในอนาคตควรพิจารณากระบวนการถ่ายโอนความรู้หรือการทดสอบกับหุ่นยนต์ขนาดเล็กก่อนนำไปใช้กับรถยนต์ขนาดจริง เพื่อยืนยันความปลอดภัยและประสิทธิภาพของอัลกอริทึมในสภาพแวดล้อมทางกายภาพที่มีข้อจำกัดด้านฮาร์ดแวร์และเซนเซอร์

บรรณานุกรม

TNI

บรรณานุกรม

- [1] S. C. K. Tekouabou et al., “Improving parking availability prediction in smart cities with IoT and ensemble-based model,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 687-697, May 2022.
- [2] G. Ali et al., “IoT based smart parking system using deep long short memory network,” *Electronics*, vol. 9, no. 10, art. no.1696, pp. 1-18, October 2020.
- [3] K. Korosec, “Daimler and Bosch’s Driverless Parking Gets OK to Operate without Human Supervision,” Tech Crunch. Blog, [Online]. Available: <https://techcrunch.com/2019/07/23/daimler-and-boschs-driverless-parking-feature-can-legally-operate-without-human-supervision/> [Accessed: February 7, 2026].
- [4] W. Kim and I. Jung, “Smart parking lot based on edge cluster computing for full self-driving vehicles,” in *IEEE Access*, vol. 10, no. 1, pp. 115271-115281, September 2022, doi: 10.1109/ACCESS.2022.3208356.
- [5] A. Uzzaman et al., “Review on the safety and sustainability of autonomous vehicles: Challenges and future directions,” *Control Systems and Optimization Letters*, vol. 3, no 1, pp. 103–109, April 2025.
- [6] Y. Deng et al., “Deep learning-based autonomous driving systems: A survey of attacks and defenses,” in *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 7897-7912, December 2021, doi: 10.1109/TII.2021.3071405.
- [7] L. Liu et al., “Computing systems for autonomous driving: State of the art and challenges,” in *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469-6486, April 2021, doi: 10.1109/JIOT.2020.3043716.
- [8] B. R. Kiran et al., “Deep reinforcement learning for autonomous driving: A survey,” in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909-4926, June 2022, doi: 10.1109/TITS.2021.3054625.
- [9] P. Ramachandran et al., “Searching for activation functions,” *6th International Conference on Learning Representations, (ICLR), Workshop*, Vancouver, BC, Canada, October 16, 2017, pp. 1-13, arXiv: 1710.05941.

- [10] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Work-Conference on Artificial and Natural Neural Networks*, (IWANN), Torremolinos, Spain, June 7, 1995, pp. 195-201, doi: 10.1007/3-540-59497-3_175.
- [11] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control*, Singapore: Springer Singapore, 2023, doi: 10.1007/978-981-19-7784-8.
- [12] R. Bellman, "Dynamic programming and stochastic control processes" *Information and Control*, vol. 1, no. 3, pp. 228-239, May 1958, doi: 10.1016/S0019-9958(58)80003-0.
- [13] P. E. Hart et al., "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.
- [14] O. Berger-Tal et al., "The exploration-exploitation dilemma: A multidisciplinary framework," *PLoS ONE*, vol. 9, no. 4, p. e95693, April 2014, doi: 10.1371/journal.pone.0095693.
- [15] S. Guo et al., "HSPOG: An optimized target recognition method based on histogram of spatial pyramid-oriented gradients," in *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 475-483, August 2021, doi: 10.26599/TST.2020.9010011.
- [16] R. Krisdianto et al., "Performance analysis of support vector machine (SVM) for diabetes disease detection," *2024 5th International Conference on Artificial Intelligence and Data Sciences, (AiDAS)*, Bangkok, Thailand, September 3, 2024, pp. 203-207, doi: 10.1109/AiDAS63860.2024.10730403.
- [17] Y. LeCun et al., "Deep learning," in *Nature*, vol. 521, no. 7553, pp. 436-444, May 2015, doi: 10.1038/nature14539.
- [18] J. Schulman et al., "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning, (ICML)*, Lille, France, July 6, 2015, pp. 1889-1897.
- [19] Q. Meng and Y. Peng, "Output-feedback quadratic tracking control of continuous-time systems by using off-policy reinforcement learning with neural networks observer," *2020 Chinese Control and Decision Conference, (CCDC)*, Hefei, China, August 22, 2020, pp. 1504-1509, doi: 10.1109/CCDC49329.2020.9164737.

- [20] A. Juliani et al., “Unity: A General Platform For Intelligent Agents,” *arXiv.org*, [Online]. Available: <https://arxiv.org/abs/1809.02627>. [Accessed: February 19, 2026].
- [21] Unity Technologies, “Training Configuration File,” Unity, [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.mlagents@4.0/manual/Training-Configuration-File.html> [Accessed: February 19, 2026].
- [22] H. Li, “Design and implement of soccer player AI training system using unity ML agents,” in *2022 3rd International Conference on Computer Information and Big Data Applications*, (CIBDA), Wuhan, China, March 25, 2022, pp. 1-4, doi:10.1109/CIBDA55480.2022.00008.
- [23] Y. Wang and Z. Yi, “Reconfigurable optical demultiplexer calling framework based on gRPC Design,” *2022 7th International Conference on Communication, Image and Signal Processing*, (CCISP), Chengdu, China, November 18, 2022, pp. 13-17, doi: 10.1109/CCISP55629.2022.9974287.
- [24] Z. Lu et al., “Towards physics-based interactive simulation of electrocautery procedures using PhysX,” *2010 IEEE Haptics Symposium*, Waltham, MA, USA, March 25, 2010, pp. 515-518, doi: 10.1109/HAPTIC.2010.5444609.
- [25] M. Toromanoff et al., “End-to-end model-free reinforcement learning for urban driving using implicit affordances,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (CVPR), Seattle, WA, USA, June 13, 2020, pp. 7151-7160, doi: 10.1109/CVPR42600.2020.00718.
- [26] J. Liao et al., “Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning,” in *IEEE Access*, vol. 8, no. 1, pp. 177804-177814, September 8, 2020, doi: 10.1109/ACCESS.2020.3022755.
- [27] Y. Guan et al., “Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization,” in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp.12597-12608, November 2020, doi:10.1109/TVT.2020.3026111.
- [28] E. Beeching et al., “Godot Reinforcement Earning Agents,” *arXiv.org*, [Online]. Available: <https://arxiv.org/abs/2112.03636>. [Accessed: February 19, 2026].

- [29] A. Singh et al., “Autonomous driving system using CNN and PPO,” *2025 3rd International Conference on Advancement in Computation & Computer Technologies*, (InCACCT), Gharuan, India, April 24, 2025, pp. 473-477, doi:10.1109/InCACCT65424.2025.11011424.
- [30] C. Xu et al., “Karting racing: A revisit to PPO and SAC algorithm,” *2021 International Conference on Computer Information Science and Artificial Intelligence*, (CISAI), Kunming, China, September 17, 2021, pp. 310-316, doi: 10.1109/CISAI54367.2021.00066.
- [31] R. Parhi and M. Unser, “Modulation spaces and the curse of dimensionality,” *2023 International Conference on Sampling Theory and Applications*, (SampTA), New Haven, CT, USA, July 10, 2023, pp. 1-5, doi: 10.1109/SampTA59647.2023.10301395.



ภาคผนวก

TNI

ภาคผนวก ก.
ชุดคำสั่งภาษา C#

TNI

โปรแกรมสำหรับระบบรางวัลของ Agent รูปแบบการให้รางวัลเมื่อถึงเป้าหมายเพียงอย่างเดียว

ชุดคำสั่ง
<pre> using System.Collections; using System.Collections.Generic; using System.Linq; using UnityEngine; using Unity.MLAgents; using Unity.MLAgents.Sensors; using Unity.MLAgents.Actuators; using UnityStandardAssets.Vehicles.Car; namespace UnityStandardAssets.Vehicles.Car { public class CarAgent : Agent { [Header("Car Movement Settings")] private Rigidbody rBody; private CarController carController; [Header("Target")] public Transform target; [Header("Spawn Area")] public List<Collider> spawnAreas = new List<Collider>(); private List<int> spawnIndexBag = new List<int>(); [Header("Sensors and Rewards")] private int parkedCorrectlyScore = 2500; private int collisionPenalty = -2000; private int stillPenalty = -2000; private int timePenalty = -10; </pre>

```
[Header("Reward Shaping")]
[Tooltip("รางวัลเมื่อเห็น Target (ขอแค่เห็น อยู่มุมไหนก็ได้)")]
public int visibleReward = 1;

[Tooltip("รางวัลเมื่อเป้าหมายอยู่กลางจอแนวนอน (Left-Right)")]
private int centeringReward = 5;

[Tooltip("รางวัลเมื่อเป้าหมายใหญ่ขึ้น (วิ่งเข้าหา)")]
private int approachReward = 5;

[Tooltip("ขอบเขตของ 'กลางจอ' แกน X (เช่น 0.1)")]
private float centerThreshold = 0.1f;

private float lastDistance;
private float[] lastActions;
private float timeSpentStill;
private float lastApparentSize = 0f;
private Collider targetCollider;
private int episodeCount = 0;

[Header("--- Statistics Table ---")]
public int successCount = 0;
public int crashCount = 0;
public int failCount = 0;

[Header("Camera Settings")]
public Camera[] carCameras;
public int cameraWidth = 84;
public int cameraHeight = 84;
public bool cameraGrayScale = false;
```

```

    public SensorCompressionType sensorCompressionType =
SensorCompressionType.PNG;

    private Camera mainCam;

    public override void Initialize()
    {
        base.Initialize();

        rBody = GetComponent<Rigidbody>();

        carController = GetComponent<CarController>();

        lastActions = new float[3];

        if (target != null)
        {
            targetCollider = target.GetComponent<Collider>();
        }

        if (carCameras.Length > 0)
        {
            mainCam = carCameras[0];
        }
    }

    public override void OnEpisodeBegin()
    {
        if (episodeCount >= 100)
        {
            float successRate = (float)successCount / episodeCount * 100f;

            Debug.Log($"<color=yellow><b>=== FINAL RESULT (100 Episodes)
===</b></color>\n" +
                $"Success: {successCount} ({successRate:F1}%)\n" +
                $"Crashed: {crashCount}\n" +

```



```

        $"Failed/Timeout: {failCount}\n" +
        $"-----");

    #if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
    #endif
    return;
}
episodeCount++;

Collider selectedSpawnArea = null;

if (spawnAreas.Count < 3)
{
    Debug.LogError("กรุณาลาก Spawn Areas ใส่ใน Inspector อย่างน้อย 3 จุด");
    return;
}

if (spawnIndexBag.Count == 0)
{
    for (int i = 0; i < spawnAreas.Count; i++)
    {
        spawnIndexBag.Add(i);
    }
    spawnIndexBag = spawnIndexBag.OrderBy(x => Random.value).ToList();
}

int randomIndex = spawnIndexBag[0];
spawnIndexBag.RemoveAt(0);
selectedSpawnArea = spawnAreas[randomIndex];

```

```

        Bounds bounds = selectedSpawnArea.bounds;
        float randomX = Random.Range(bounds.min.x, bounds.max.x);
        float randomZ = Random.Range(bounds.min.z, bounds.max.z);
        this.transform.position = new Vector3(randomX, this.transform.position.y,
        randomZ);

        float spawnZoneYRotation = selectedSpawnArea.transform.eulerAngles.y;
        float randomRotationOffset = Random.Range(-20f, 20f);
        float finalYRotation = spawnZoneYRotation + randomRotationOffset;
        this.transform.rotation = Quaternion.Euler(0f, finalYRotation, 0f);

        rBody.velocity = Vector3.zero;
        rBody.angularVelocity = Vector3.zero;

        lastDistance = Vector3.Distance(transform.position, target.position);
        lastApparentSize = 0f;
        System.Array.Clear(lastActions, 0, lastActions.Length);
        timeSpentStill = 0f;
    }

    public override void CollectObservations(VectorSensor sensor)
    {
        Vector3 localTargetPos = transform.InverseTransformPoint(target.position);
        sensor.AddObservation(localTargetPos.x / 100f);
        sensor.AddObservation(localTargetPos.z / 100f);

        float angleToTarget = Vector3.SignedAngle(transform.forward, (target.position
        - transform.position), Vector3.up);
        sensor.AddObservation(angleToTarget / 180f);

        Vector3 localVelocity = transform.InverseTransformDirection(rBody.velocity);

```

```
sensor.AddObservation(localVelocity.x / (carController.MaxSpeed + 1e-6f));
sensor.AddObservation(localVelocity.z / (carController.MaxSpeed + 1e-6f));

sensor.AddObservation(lastActions);
if (mainCam != null && target != null)
{
    Vector3 viewPos = mainCam.WorldToViewportPoint(target.position);

    bool isInFront = viewPos.z > 0;

    if (isInFront)
    {
        float xDeviation = viewPos.x - 0.5f;
        float yDeviation = viewPos.y - 0.5f;

        sensor.AddObservation(xDeviation);
        sensor.AddObservation(yDeviation);
        sensor.AddObservation(1.0f);
    }
    else
    {
        sensor.AddObservation(0f);
        sensor.AddObservation(0f);
        sensor.AddObservation(0f);
    }
}
else
{
    sensor.AddObservation(0f);
    sensor.AddObservation(0f);
    sensor.AddObservation(0f);
}
```

```

    }
}

public override void OnActionReceived(ActionBuffers actions)
{
    var continuousActions = actions.ContinuousActions;
    float steering = continuousActions[0];
    float accel = continuousActions[1];
    float reverse = continuousActions[2];
    float finalAccel = Mathf.Clamp(accel - reverse, -1f, 1f);

    carController.Move(steering, finalAccel, 0f, 0f);
    float currentApparentSize = 0f;

    bool isVisible = false;
    bool isCenteredX = false;
    bool isApproaching = false;
    float growth = 0f;
    AddReward(timePenalty);

    float currentDistance = Vector3.Distance(transform.position, target.position);
    float DistanceReward = -((currentDistance) * 1.5f) - 25;

    if (carCameras.Length > 0 && targetCollider != null)
    {
        Camera mainCamera = carCameras[0];

        Vector3 targetTop = targetCollider.bounds.max;
        Vector3 targetBottom = targetCollider.bounds.min;
        Vector3 topScreen = mainCamera.WorldToViewportPoint(targetTop);
        Vector3 botScreen = mainCamera.WorldToViewportPoint(targetBottom);
    }
}

```



```

if (topScreen.z > 0 && botScreen.z > 0)
{
    currentApparentSize = Mathf.Abs(topScreen.y - botScreen.y);

    Vector3 centerScreen =
mainCamera.WorldToViewportPoint(target.position);

    bool isOnScreen = centerScreen.x >= 0 && centerScreen.x <= 1 &&
        centerScreen.y >= 0 && centerScreen.y <= 1;

    Vector3 localVel = transform.InverseTransformDirection(rBody.velocity);
    bool isMovingForward = localVel.z > 1.0f;

    if (isOnScreen)
    {
        float distX = Mathf.Abs(centerScreen.x - 0.5f);
        if (distX < centerThreshold)
        {
            isCenteredX = true;
        }
    }
    if (isMovingForward && (currentApparentSize > lastApparentSize))
    {
        isApproaching = true;
        growth = currentApparentSize - lastApparentSize;
    }

    if (isOnScreen)
    {
        isVisible = true;
        float distX = Mathf.Abs(centerScreen.x - 0.5f);

```

```

        if (distX < centerThreshold)
        {
            isCenteredX = true;
        }
    }
}

if (isVisible)
{
    AddReward(visibleReward);
    Debug.Log($"<color=lime>[VISIBLE] Target in view! ({visibleReward})</color>");
}
if (isCenteredX)
{
    AddReward(centeringReward);
    Debug.Log($"<color=cyan>[CENTER] X-Centered! ({centeringReward})</color>");
}
if (isApproaching)
{
    AddReward(approachReward);
    Debug.Log($"<color=green>[APPROACH] Bigger! Growth: {growth:F6}
    ({approachReward})</color>");
}
if (mainCam != null && target != null)
{
    Vector3 viewPos = mainCam.WorldToViewportPoint(target.position);
    if (viewPos.z > 0 && viewPos.x >= 0 && viewPos.x <= 1 && viewPos.y >=
0 && viewPos.y <= 1)
    {
        float distFromCenter = Mathf.Abs(viewPos.x - 0.5f);
        float aimReward = 1.0f - (distFromCenter * 2.0f);
    }
}

```

```

        Debug.Log($"Aim Accuracy: {aimReward * 100}%");
    }
}

lastApparentSize = currentApparentSize;
lastActions[0] = steering;
lastActions[1] = accel;
lastActions[2] = reverse;
}

public override void Heuristic(in ActionBuffers actionsOut)
{
    var continuousActionsOut = actionsOut.ContinuousActions;
    continuousActionsOut[0] = Input.GetAxis("Horizontal");
    continuousActionsOut[1] = Input.GetAxis("Acceleration");
    continuousActionsOut[2] = Input.GetAxis("Reverse");
}

public void FixedUpdate()
{
    if (rBody.velocity.magnitude < 0.1f)
    {
        timeSpentStill += Time.fixedDeltaTime;
        if (timeSpentStill > 5.0f)
        {
            failCount++;
            AddReward(stillPenalty);
            Debug.Log($"<color=red>Episode {episodeCount} Ended (Still). Total  

            Reward: {GetCumulativeReward():F2}</color>");
            EndEpisode();
        }
    }
}

```

```

    }
    else
    {
        timeSpentStill = 0f;
    }
}

private void OnCollisionEnter(Collision other)
{
    if (other.gameObject.CompareTag("Wall") || other.gameObject.CompareTag("Car"))
    {
        crashCount++;
        AddReward(collisionPenalty);
        Debug.Log($"<color=red>Episode {episodeCount} Ended (Collision). Total
Reward: {GetCumulativeReward():F2}</color>");
        EndEpisode();
    }
}

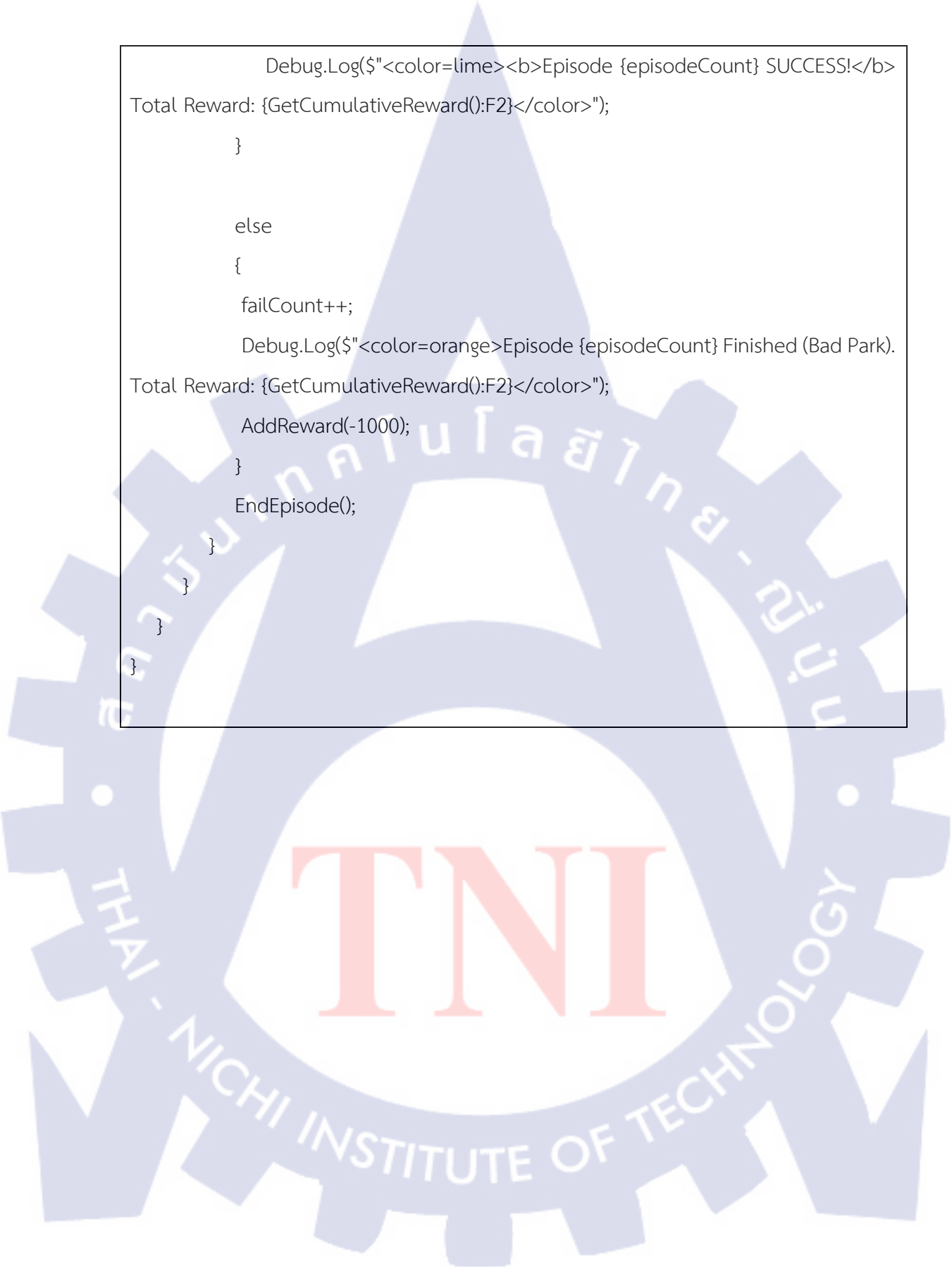
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Finish"))
    {
        float currentAngle = Vector3.Angle(transform.forward, target.forward);
        if (rBody.velocity.magnitude <= 3.0f && (currentAngle < 10f || currentAngle
> 170f))
        {
            successCount++;
            AddReward(parkedCorrectlyScore);

```



```
        Debug.Log($"<color=lime><b>Episode {episodeCount} SUCCESS!</b>
Total Reward: {GetCumulativeReward():F2}</color>");
    }

    else
    {
        failCount++;
        Debug.Log($"<color=orange>Episode {episodeCount} Finished (Bad Park).
Total Reward: {GetCumulativeReward():F2}</color>");
        AddReward(-1000);
    }
    EndEpisode();
}
}
```



THAI-NICHI INSTITUTE OF TECHNOLOGY

TNI

โปรแกรมสำหรับระบบรางวัลของ Agent รูปแบบการให้รางวัลมีแรงจูงใจให้ Agent

ชุดคำสั่ง

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Sensors;
using Unity.MLAgents.Actuators;
using UnityStandardAssets.Vehicles.Car;

namespace UnityStandardAssets.Vehicles.Car
{
    public class CarAgent : Agent
    {
        [Header("Car Movement Settings")]
        private Rigidbody rBody;
        private CarController carController;

        [Header("Target")]
        public Transform target;

        [Header("Spawn Area")]
        public List<Collider> spawnAreas = new List<Collider>();
        private List<int> spawnIndexBag = new List<int>();

        [Header("Sensors and Rewards")]
        private int parkedCorrectlyScore = 2500;
        private int collisionPenalty = -2000;
        private int stillPenalty = -2000;
        private int timePenalty = -10;
```

```

[Header("Reward Shaping")]
[Tooltip("รางวัลเมื่อเห็น Target (ขอแค่เห็น อยู่มุมไหนก็ได้)")]
public int visibleReward = 1;
[Tooltip("รางวัลเมื่อเป้าหมายอยู่กลางจอแนวนอน (Left-Right)")]
private int centeringReward = 5;
[Tooltip("รางวัลเมื่อเป้าหมายใหญ่ขึ้น (วิ่งเข้าหา)")]
private int approachReward = 5;

[Tooltip("ขอบเขตของ 'กลางจอ' แกน X (เช่น 0.1)")]
private float centerThreshold = 0.1f;

private float lastDistance;
private float lastAngle;
private float[] lastActions;
private float timeSpentStill;

private float lastApparentSize = 0f;
private Collider targetCollider;
// -----

private int episodeCount = 0;

[Header("--- Statistics Table ---")]
public int successCount = 0;
public int crashCount = 0;
public int failCount = 0;

[Header("Camera Settings")]
public Camera[] carCameras;
public int cameraWidth = 84;

```

```

public int cameraHeight = 84;
public bool cameraGrayScale = false;
public SensorCompressionType sensorCompressionType =
SensorCompressionType.PNG;

public override void Initialize()
{
    rBody = GetComponent<Rigidbody>();
    carController = GetComponent<CarController>();
    lastActions = new float[3];

    if (target != null)
    {
        targetCollider = target.GetComponent<Collider>();
    }
}

public override void OnEpisodeBegin()
{
    if (episodeCount >= 100)
    {
        float successRate = (float)successCount / episodeCount * 100f;

        Debug.Log($"<color=yellow><b>=== FINAL RESULT (100 Episodes)
===</b></color>\n" +
            $"Success: {successCount} ({successRate:F1}%)\n" +
            $"Crashed: {crashCount}\n" +
            $"Failed/Timeout: {failCount}\n" +
            $"-----");
    }
}

#if UNITY_EDITOR

```



```

        UnityEditor.EditorApplication.isPlaying = false;
    #endif
    return;
}

episodeCount++;
Collider selectedSpawnArea = null;

if (spawnAreas.Count < 3)
{
    Debug.LogError("กรุณาลาก Spawn Areas ใส่ใน Inspector อย่างน้อย 3 จุด");
    return;
}

if (spawnIndexBag.Count == 0)
{
    for (int i = 0; i < spawnAreas.Count; i++)
    {
        spawnIndexBag.Add(i);
    }

    spawnIndexBag = spawnIndexBag.OrderBy(x => Random.value).ToList();
}

int randomIndex = spawnIndexBag[0];
spawnIndexBag.RemoveAt(0);
selectedSpawnArea = spawnAreas[randomIndex];

Bounds bounds = selectedSpawnArea.bounds;
float randomX = Random.Range(bounds.min.x, bounds.max.x);
float randomZ = Random.Range(bounds.min.z, bounds.max.z);

```

```

        this.transform.position = new Vector3(randomX, this.transform.position.y,
        randomZ);

        float spawnZoneYRotation = selectedSpawnArea.transform.eulerAngles.y;
        float randomRotationOffset = Random.Range(-20f, 20f);
        float finalYRotation = spawnZoneYRotation + randomRotationOffset;
        this.transform.rotation = Quaternion.Euler(0f, finalYRotation, 0f);

        rBody.velocity = Vector3.zero;
        rBody.angularVelocity = Vector3.zero;

        lastDistance = Vector3.Distance(transform.position, target.position);
        lastApparentSize = 0f;
        System.Array.Clear(lastActions, 0, lastActions.Length);
        timeSpentStill = 0f;
    }

    public override void CollectObservations(VectorSensor sensor)
    {
        Vector3 localTargetPos = transform.InverseTransformPoint(target.position);
        sensor.AddObservation(localTargetPos.x / 100f);
        sensor.AddObservation(localTargetPos.z / 100f);

        float angleToTarget = Vector3.SignedAngle(transform.forward,
        (target.position - transform.position), Vector3.up);
        sensor.AddObservation(angleToTarget / 180f);

        Vector3 localVelocity =
        transform.InverseTransformDirection(rBody.velocity);
        sensor.AddObservation(localVelocity.x / (carController.MaxSpeed + 1e-6f));
        sensor.AddObservation(localVelocity.z / (carController.MaxSpeed + 1e-6f));
    }

```

```

        sensor.AddObservation(lastActions);
    }

    public override void OnActionReceived(ActionBuffers actions)
    {
        var continuousActions = actions.ContinuousActions;
        float steering = continuousActions[0];
        float accel = continuousActions[1];
        float reverse = continuousActions[2];
        float finalAccel = Mathf.Clamp(accel - reverse, -1f, 1f);

        carController.Move(steering, finalAccel, 0f, 0f);

        float currentDistance = Vector3.Distance(transform.position, target.position);
        float directionDot = Vector3.Dot(transform.forward.normalized,
            (target.position - transform.position).normalized);

        float velocityDot = (rBody.velocity.magnitude > 0.1f) ?
            Vector3.Dot(rBody.velocity.normalized, (target.position -
            transform.position).normalized) : 0f;

        float currentAngle = Vector3.Angle(transform.forward, (target.position -
            transform.position));

        // --- Reward Shaping ---
        AddReward((lastDistance - currentDistance) * 2.0f);
        AddReward(directionDot * 0.1f);
        AddReward(velocityDot * 0.2f);
        AddReward(timePenalty);
        AddReward(Mathf.Cos(currentAngle * Mathf.Deg2Rad) * 0.3f);
        if (currentDistance < 5f)
        {

```

```

        AddReward(2f * (5f - currentDistance));
    }
    if (currentDistance < 10f)
    {
        AddReward(0.1f);
    }
    if (currentDistance > lastDistance + 0.1f)
    {
        AddReward(-0.2f);
    }
    float currentApparentSize = 0f;

    bool isVisible = false;
    bool isCenteredX = false;
    bool isApproaching = false;
    float growth = 0f;

    float Distance = Vector3.Distance(transform.position, target.position);
    float DistanceReward = -((Distance) * 3) - 10;
    AddReward(DistanceReward);
    Debug.Log($"<color=lime>Distance Reward = {DistanceReward}</color>");

    if (carCameras.Length > 0 && targetCollider != null)
    {
        Camera mainCamera = carCameras[0];

        Vector3 targetTop = targetCollider.bounds.max;
        Vector3 targetBottom = targetCollider.bounds.min;
        Vector3 topScreen = mainCamera.WorldToViewportPoint(targetTop);
        Vector3 botScreen = mainCamera.WorldToViewportPoint(targetBottom);
    }

```

```

if (topScreen.z > 0 && botScreen.z > 0)
{
    currentApparentSize = Mathf.Abs(topScreen.y - botScreen.y);
    Vector3 centerScreen =
mainCamera.WorldToViewportPoint(target.position);

    bool isOnScreen = centerScreen.x >= 0 && centerScreen.x <= 1 &&
        centerScreen.y >= 0 && centerScreen.y <= 1;

    Vector3 localVel =
transform.InverseTransformDirection(rBody.velocity);
    bool isMovingForward = localVel.z > 1.0f;

    if (isOnScreen)
    {
        float distX = Mathf.Abs(centerScreen.x - 0.5f);
        if (distX < centerThreshold)
        {
            isCenteredX = true;
        }
    }
    if (isMovingForward && (currentApparentSize > lastApparentSize))
    {
        isApproaching = true;
        growth = currentApparentSize - lastApparentSize;
    }

    if (isOnScreen)
    {
        isVisible = true;
        float distX = Mathf.Abs(centerScreen.x - 0.5f);

```



```
        if (distX < centerThreshold)
        {
            isCenteredX = true;
        }
    }
}

if (isVisible)
{
    //AddReward(visibleReward);
    Debug.Log($"<color=lime>[VISIBLE] Target in view!
    ({visibleReward})</color>");
}

if (isCenteredX)
{
    //AddReward(centeringReward);
    Debug.Log($"<color=cyan>[CENTER] X-Centered!
    ({centeringReward})</color>");
}

if (isApproaching)
{
    AddReward(approachReward);
    Debug.Log($"<color=green>[APPROACH] Bigger! Growth: {growth:F6}
    ({approachReward})</color>");
}

lastApparentSize = currentApparentSize;
lastAngle = currentAngle;
```

```

lastDistance = currentDistance;
lastActions[0] = steering;
lastActions[1] = accel;
lastActions[2] = reverse;
}

public override void Heuristic(in ActionBuffers actionsOut)
{
    var continuousActionsOut = actionsOut.ContinuousActions;
    continuousActionsOut[0] = Input.GetAxis("Horizontal");
    continuousActionsOut[1] = Input.GetAxis("Acceleration");
    continuousActionsOut[2] = Input.GetAxis("Reverse");
}

public void FixedUpdate()
{
    if (rBody.velocity.magnitude < 0.1f)
    {
        timeSpentStill += Time.fixedDeltaTime;

        if (timeSpentStill > 5.0f)
        {
            AddReward(stillPenalty);
            Debug.Log($"<color=red>Episode {episodeCount} stillPenalty Score:
{stillPenalty}</color>");
            EndEpisode();
        }
    }
    else
    {
        timeSpentStill = 0f;
    }
}

```

```

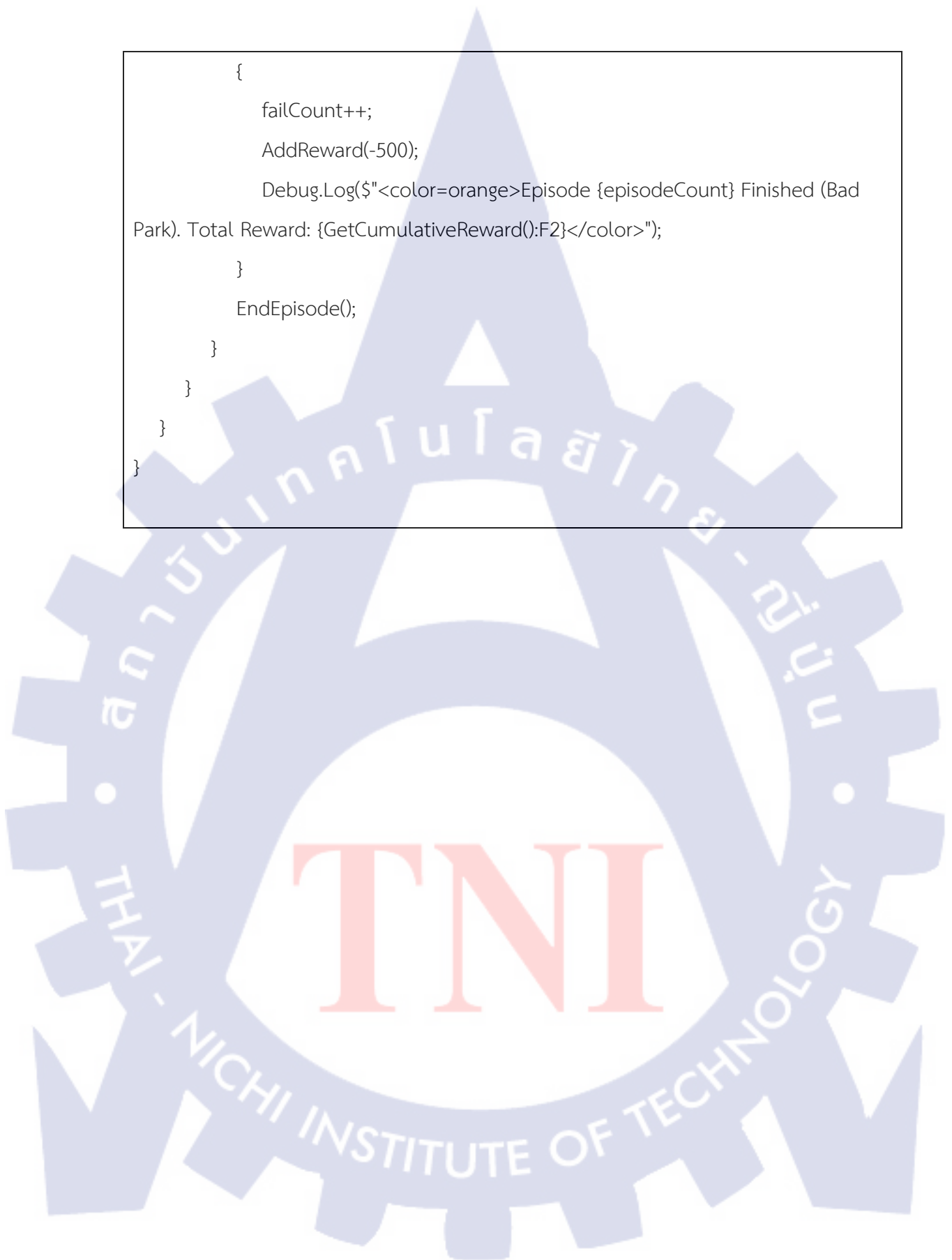
    }
}

private void OnCollisionEnter(Collision other)
{
    if (other.gameObject.CompareTag("Wall") ||
other.gameObject.CompareTag("Car"))
    {
        crashCount++;
        //AddReward(collisionPenalty);
        Debug.Log($"<color=red>Episode {episodeCount} Ended (Collision).
Total Reward: {GetCumulativeReward():F2}</color>");
        EndEpisode();
    }
}

private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Finish"))
    {
        float currentAngle = Vector3.Angle(transform.forward, target.forward);
        if (rBody.velocity.magnitude <= 5.0f && (currentAngle < 20f ||
currentAngle > 160f))
        {
            successCount++;
            AddReward(parkedCorrectlyScore);
            Debug.Log($"<color=lime><b>Episode {episodeCount} SUCCESS!</b>
Total Reward: {GetCumulativeReward():F2}</color>");
        }
        else

```

```
{  
    failCount++;  
    AddReward(-500);  
    Debug.Log($"<color=orange>Episode {episodeCount} Finished (Bad  
Park). Total Reward: {GetCumulativeReward():F2}</color>");  
}  
EndEpisode();  
}  
}  
}  
}
```



ภาคผนวก ข.
บทความการประชุมวิชาการ

TNI

Game-Based Autonomous Parking System using Proximal Policy Optimization with Reward Shaping

Vorrapong Srisaard
Master Program in Engineering
Technology
Thai Nichi Institute of Technology
Bangkok, Thailand
2461110039@tni.ac.th

Verapong Srisaard
Department of Computer Engineering and
Artificial Intelligence
Thai Nichi Institute of Technology
Bangkok, Thailand
sr.verapong_st@tni.ac.th

Sivapong Nilwong
Advanced Computation Systems Research
Laboratory
Thai Nichi Institute of Technology
Bangkok, Thailand
sivapong@tni.ac.th

Abstract— This paper presents an autonomous parking system for self-driving cars using information from a game-based simulation. Proximal Policy Optimization (PPO) algorithm is used as the main algorithm of the parking system. Visual information from cameras on the front and rear of the car, and the observed states of the car are used as the input. The agents for the parking system are trained in the simulation environment powered by the Unity game engine. Two different agents are trained using the PPO with different rewarding systems. The first agent receives rewards only when the agent reaches the goal. The second agent is trained with the reward shaping technique. Additional rewards are granted to the second agent in every action step during the training process. Agents are trained and tested in the simulation scenario where the agent must reach the parking slot in the correct orientation from random starting locations. Experiments include car parking tests of the agents in the simulation. Success rate of correctly parking a car at the goal location is used to evaluate parking performances of the agents. Simulation results show superior performance of the agent trained with the reward shaping technique in the task of parking a car.

Keywords—autonomous parking, proximal policy optimization, reward shaping, unity, deep reinforcement learning.

I. INTRODUCTION

In present world, about 30% of city traffic problems come from vehicles searching for parking spots [1]. Drivers spend a long time looking for parking spots, in which fuel is wasted, causing pollution [2]. One of available solutions to car parking problems is the implementation of self-driving vehicles [3]. Different models of Artificial Intelligence (AI) are applied in most self-driving vehicles, especially self-driving cars, to provide better experiences for vehicle users [4]. However, applications of AI models on self-driving cars, encounter different difficulties. Some examples of such difficulties include unpredictable environment [5], vulnerability to cyberattacks [6], and resource-intensive [7].

Among AI models and algorithms available for self-driving cars, algorithms which applied Reinforcement Learning (RL) accomplished significant achievements in recent findings. The abilities in learning through trial-and-error approaches of RL algorithms displayed their performances in many works such as [8], [9], and [10]. One notable RL algorithm is the Proximal Policy Optimization (PPO), which trains its agents based on policy functions. PPO has been the default RL algorithm for OpenAI since 2018 [11]. Performances of PPO were evaluated in different works such as [12], and [13].

From satisfying outcomes of PPO applications in various visual-based applications, self-driving cars can also be implemented with the PPO. Same as other RL algorithms, simulation environments which resembles the real applications are required to train agents with PPO. Therefore, this paper presents the autonomous parking system for self-driving car using visual information with the PPO in the simulated world. The agent which represents the car applies a camera as its only sensor. The parking algorithm is learned through trial-and-error approach in the PPO instead of hard-coding. Unity game engine is implemented as the basis for the simulation of the agent in both training and testing. The implementation of the Unity engine is due to its simplicity and resources available [14]. Two PPO agents are developed and tested for the car parking task, in which the agents need to move forward to the parking slot from random starting locations within the limited time. The agents possess different reward functions during the training via PPO. One agent acquires a big reward when it reaches its parking slot. Another agent is trained with the reward shaping technique, in which rewards are granted to the agent in every action. The second agent also receives a big reward once it reaches the parking slot. Agents for the parking system are tested in the simulation environment with the same settings as in the training. Experiment results show satisfying results from both agents in reaching the parking slot that located in front of the car, within 45 degrees of orientations from the random starting locations. Additionally, the agent trained with the reward shaping technique performs better in the car parking experiments.

This paper is organized as follows: Section 2 describes the proposed parking system and its components, Section 3 explains the experiments and provides experimental results, and Section 4 concludes the contents and discuss future works.

II. AUTONOMOUS PARKING SYSTEM

The proposed autonomous parking system has its workflow as depicts in Fig. 1. The system takes images from the front and the rear cameras as its input. These images are fed directly to the core computation component which is a model based on the proximal policy optimization (PPO) algorithm. The model controls the car, guiding the car to the parking area using knowledge acquired from the training process in the simulation environment. Further details of the implemented simulation and the PPO models are described in following subsections.

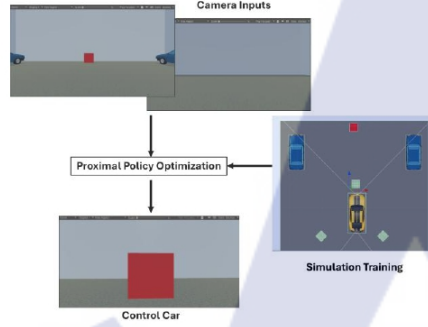


Fig. 1. Diagram of the game-based autonomous parking system.

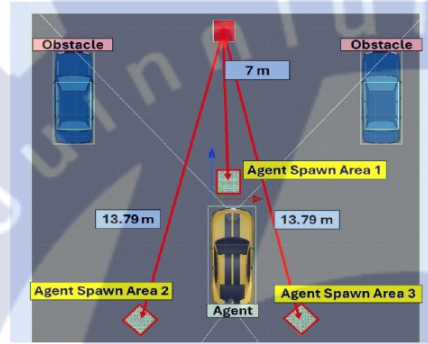


Fig. 2. Top view of the simulation environment.

A. Unity Simulation Environment

The simulation environment is created using Unity. Unity is a popular platform for developing 2D and 3D games and simulations. There are many resources and assets available in Unity platform, which ease the development of the simulation environments. From the top view of the simulation in Fig. 2, the simulation environment is a simple square room with white walls on all its four sides. This simulation environment is utilized in both the training and the experiments of the autonomous parking system. Different assets from the Unity engine are applied as objects in the developed simulation environment. Blue cars are used as obstacles. The yellow car is employed as the car for the agent. A red cube is placed as the goal, which determines the parking area. There are three starting areas for the agent, i.e. spawn areas. The first spawn area is closer to the goal, with the spawn orientation of the area facing the goal. The other two areas have spawn orientations angled 45 degrees to the goal, with longer distances to the goal compared to the first area.

The navigation task in the simulation requires the agent to reach the goal which is located within short distances from the starting point. The agent must be able to park the car at the specific location, within the acceptable orientations. The

TABLE I. CAR PARKING SIMULATION SETTINGS

Simulation Properties	Applied Settings
Time Limit per Episode	5 seconds
Camera Inputs	$84 \times 84 \times 1$ (Grayscale Image)
Agent Actions	4

TABLE II. BASIC SIMULATION REWARDS

Agent State	Provide Rewards
Time Usage	- 10
Dead	- 2,000
Bad Park	- 1,000
Goal	2,500
Distance Penalty	$-(Distance_{Goal} \times 1.5) - 25$
Camera Reward	+ 5
Center Reward	$1 - (image_x_{Goal} - image_x_{center} \times 2)$

actual spawn point and initial orientation of the car agent are randomized within the three spawn areas at the start of each episode. The goal and obstacle cars are fixed in their places. The agent must reach the goal within 5 seconds, as the simulation episode ends immediately afterwards. The episode can also end before the timer reaches 5 seconds by having the agent reach the goal, or the agent gets terminated by crashing into walls or obstacles.

Table 1 shows implemented simulation settings. Each episode has a time limit of 5 seconds. Visual information from front and back cameras of the car is in the form of RGB frames, with the size of 84×84 pixels. Each of these RGB frames are converted to a grayscale image before passing to the agent. Agent can commit 4 actions to the car including moving forward, moving backward, turning left, and turning right, resembling the simplified actions of actual cars.

Similar to other RL algorithms, agents training through the PPO algorithm require rewards to motivate them. The simulation provides base rewards as described in Table 2. The agent receives a -10 reduction on the total rewards in each step it takes, making the agent move to the goal as quickly as possible. The largest negative reward of -2,000 is given to the agent if the agent gets terminated from crashing. Another larger negative reward of -1,000 is also given to the agent if the agent reaches the goal with the wrong orientation, without straightly facing the goal. A minor penalty is given to the agent based on its location. The greater the distance between the agent and the goal, the greater this penalty will be. Apart from negative rewards, the simulation gives different positive rewards based on desirable outcomes of the actions. The largest positive reward of 2,500 is given to the agent when the car reaches the parking slot within acceptable orientations, i.e. facing the goal. The reward of 5 is given to the agent on each step that has the goal within its input image from the front camera. A reward is also given to the agent based on the distance between the reward and the center of the image from the front camera. The closer to the center the goal is, the more positive this reward becomes.

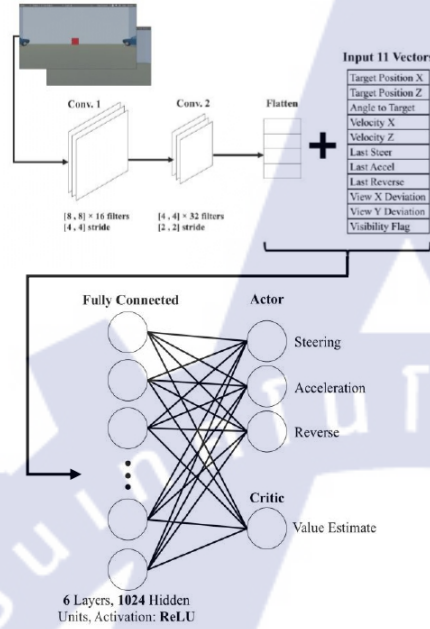


Fig. 3. Structure of the PPO for autonomous car parking agents.

B. Proximal Policy Optimization for Autonomous Parking

Proximal Policy Optimization (PPO) is applied to train agents for the task of parking in the simulation environment. The implemented PPO model has its structure as shown in Fig. 3. This PPO model is applied in the training of both the normal agent and the agent with reward shaping. Visual information from cameras attached to the car and the observed state of the car are used as the input of the PPO. Visual frames from the first-person perspective of the front and back cameras are in the form of 84×84 RGB images. These images are processed into grayscale images before passing through the convolutional layers of the PPO. Result feature maps from convolutional layers are then flattened into a 1-directional array. The array of flattened feature maps is combined with observation space vectors to be used as the input for the fully connected layers further. Unity engine offers an 11-vector observation space for PPO training. These 11 vector observations comprise target position in X and Z axes, angle to the target, velocity in X and Z axes, last steer, last accel, last reverse, view deviations in X and Y, and the visibility flag. The flattened feature maps and observation vectors are concatenated to form a state representation. The state is then fed to the fully connected layers of the PPO. The fully connected part of the PPO consists of 6 fully connected layers, with 1024 hidden units in each layer. All the fully connected layers in the implemented PPO utilize the ReLU as the activation function. The output layer of the PPO with

TABLE III PPO AGENTS TRAINING CONFIGURATIONS

Parameter	Value
trainer_type	ppo
max_steps	5,000,000
learning_rate	0.0001
learning_rate_schedule	linear
beta	0.001
epsilon	0.2
lambda	0.95
batch_size	4,096
buffer_size	40,960
num_epoch	15
gamma	0.999

4 units are separated into two sections: Actor and Critic. The actor section, or the Actor Head, applies three units from the output as actions for the car agent. The actions include steering, acceleration, and reverse. The critic section or the Critic Head applies a single unit from the output as the “Value Estimate” variable for computing the advantage function. As a result, the output of two sectors enables the parallel optimization of both policy and value estimation of the PPO.

Configurations for the training of both normal PPO agent and the PPO agent with reward shaping are stated in Table 3. These training configurations, which contain hyperparameters and settings for training the PPO agents, are provided by the Unity engine. Parameters of the training configurations can be edited to suit the tasks of agents [15]. The number of steps in the training is set to 5,000,000 steps. The initial learning rate is set to 0.0001, which decays linearly to 0 as the training gradually reaches the last step. The beta parameter that can cause the policy to become more random is set to 0.001, which is lower than the default value of 0.005. The epsilon parameter for evolving the policy and the “lambda” or the lambda for regularization remain at the default values of 0.2 and 0.95, respectively. The batch size of the experiences in each iteration of gradient descent is set to 4,096. The buffer size is set to 40,960, which is ten times larger than the batch size. The buffer size of 40,960 means that the PPO collects 40,960 experiences before updating the policy model. The “num_epoch” parameter, which defines the number of iterations that the PPO iterates over the collected experience buffer, is set to 15. The reward discount factor or the gamma parameter is set to 0.999. The remaining parameters for the PPO training from [15] which are not mentioned in Table 3, remain at their default values.

The training of the PPO agents for car parking takes place in the simulation environment as described in the previous subsection. The goal of the car parking agents is to reach the goal location, which refers to the parking slot, in the acceptable orientations. The car of the agent starts randomly across the three agent spawn areas in each simulation episode. The starting orientation of the car is also random, as influenced by the agent spawn area. The agent needs to move the car from random starting points with random orientations

TABLE IV. ADDITIONAL REWARDS FOR REWARD SHAPING AGENT

Agent State	Provide Rewards
Approach Target	$(Distance_{step} - Distance_{step-1}) \times 2$
Face Right Direction	0.1
Drive Right Direction	0.2
Angle Alignment	$\cos(\theta_{goal}) \times 0.3$
Within Range ($< 10m$)	0.1
Proximity ($< 5m$)	$2 \times (5 - Distance_{step})$
Moving Away	-0.2

to the parking location as quickly as possible. The first PPO agent was trained in the parking task, receiving a basic reward from the simulation as shown in Table 2. The second PPO agent was also trained in the same task, receiving the same basic reward as the first. However, the second PPO agent was trained with a reward customization scheme, which awards additional rewards to the agent for each action. The additional rewards for the customized agent are shown in Table 4. Rewards are added when the vehicle approaches the target, calculated by multiplying the distance from the target in the current and previous action steps by 2. Additional rewards are added when the vehicle is facing directly towards the target and turning towards it, awarding 0.1 and 0.2 respectively. The angle of the vehicle to the target is also calculated and added to the reward, using the $\cos(s)$ function of the angle to the target multiplied by 0.3. In terms of distance, a positive 0.1 reward is added when the vehicle is within 10 meters of the target. Furthermore, another reward is added, calculated from the agent's current distance to the target, when the vehicle moves within 5 meters of the target. However, the agent receives a negative -0.2 reward for actions that cause the vehicle to move away from the target. For each action performed by the agent, the additional rewards from Table 4 are added and awarded based on the agent's status after that action. Values of the additional rewards were chosen on a trial-and-error basis. Higher multipliers on the rewards of a given state signify higher priority for the state, and vice versa.

Both agents were trained for 5,000,000 steps separately. Fig. 4 shows the cumulative rewards from the simulation, which were given to the first agent with basic simulation rewards. In the first 1,000,000 steps, the rewards tend to be more random. The cumulative rewards steadily increased more than 50 times from the step 1,000,000 to 2,000,000. The rewards became more saturated from the step 2,000,000 onwards. These characteristics of the rewards can be implied that actions of the agent prior to the step 1,000,000 were most likely random, which resulted in low cumulative rewards. As the training proceeded to the step 2,000,000, the agent gradually selected actions that resulted in more satisfying outcomes from the simulation, which can be observed from the great increments of cumulative rewards during these training steps. After training for 2,000,000 steps, the agent successfully discovered suitable actions for the states of the car, as suggested by the saturated rewards from the step 2,000,000 to the final step of 5,000,000 in the training.

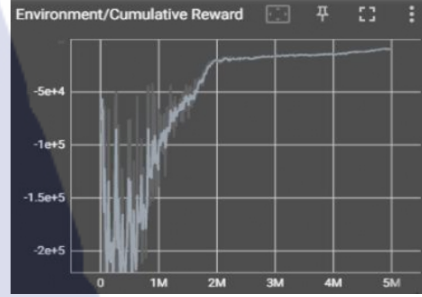


Fig. 4. Cumulative training rewards from the agent with basic rewards.

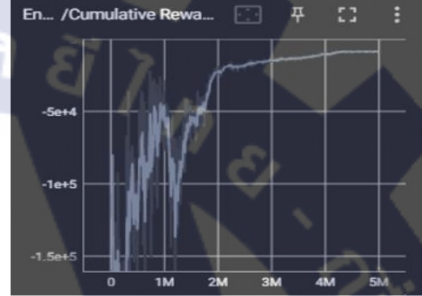


Fig. 5. Cumulative training rewards from the agent with reward shaping.

The training of the agent with reward shaping resulted in different characteristics in the cumulative rewards, as shown in Fig. 5. The cumulative rewards prior to the training step of 1,000,000 were an increasing trend, which suggested that the additional rewards helped the agent find a suitable policy for the task. However, the cumulative rewards sharply decreased shortly after the step 1,000,000, then quickly recovered and increased afterwards. This high fluctuation in the rewards was suspected to be caused by the discovery process of the agent, which had the settings of parameters related to the discovery and random actions set as in Table 3. Apart from the fluctuation, the rewards greatly increased until the training step 2,000,000, then slightly saturated to the end of the training. This can be implied that the agent with reward shaping was successfully discovering the best actions to complete the task prior to the step 2,000,000, then suitable actions were discovered afterwards. One notable point is that the agent with reward shaping achieved higher cumulative rewards in majority of the training steps.

III. EXPERIMENTS AND RESULTS

The trained PPO agents were tested for the task of parking in the simulation which possessed the same settings as in the training. Each agent was tested with the parking tasks 100 times in a 3D simulated environment. The agent started



Fig. 6. Example of the agent initial states in car parking experiments.

TABLE V. EXPERIMENTAL RESULTS OF THE CAR PARKING AGENTS

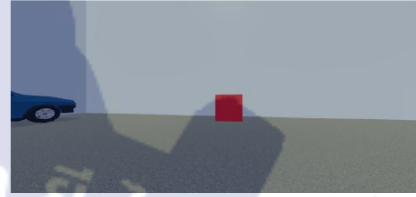
Measurements	Basic Rewards Agent	Reward Shaping Agent
Number of Experiments	100	100
Success Count	69	98
Success Rate	69 %	98 %
Crash Count	1	0
Crash Rate	1 %	0 %
Time Out	30	2
Time-Out Rate	30 %	2 %
Maximum Distance	13.79 m	13.79 m

randomly in the starting areas or the “spawn areas” indicated in Fig. 2. An example of the initial state of the car is shown in Fig. 6. From the starting location, the agent needs to move the car to the red cube goal in acceptable orientations. Fig. 7 illustrates the first-person perspective from the front camera of the car. The agent must move the car from the random starting location, as an example shown in Fig. 7(a). The agent needs to steer the car towards the goal while also moving the car closer to the goal, as the goal object becomes larger and closer to the center of image in Fig. 7(b). The parking is completed once the car reaches the goal within the acceptable orientations, as shown in Fig. 7(c). The car needed to touch the goal, with the goal object near the center of the image from the front camera of the car as much as possible.

Results from 100 testing episodes of both agents are described in Table 5. From 100 experimental episodes, the agent with basic rewards training successfully parked the car 69 times, achieved the success rate of 69%. Among the 31 failed attempts, the agent crashed 1 time, which counts as 1% crash rate. The remaining failures occurred by the agent not being able to move to the goal location within the time limit of 5 seconds. The agent reached the time limit 30 times, counting as 30% time-out rate. The agent with the reward shaping achieved a better success rate, as the agent was able to reach the goal within the time limit in 98 out of 100 experiments. The reward shaping agent failed only 2 times in the experiments, which translates to 2% failure rate. The failed attempts of the agent were caused by the time limit only, as the agent never crashed.



(a)



(b)



(c)

Fig. 7. Images from the front camera of the car in different states: (a) initial state, (b) moving towards the goal, (c) reaching the goal.

The experimental results showed that the difference in the success rate is 29%, with the reward shaping agent superior to the basic agent. From experiments, an introduction of the reward shaping during the agent training resulted in an increment of approximately 42% in the parking success rate. Details on the failure rates can be implied that the agent with basic rewards tended to stay still to minimize the penalties, once the agent reached the states which were difficult to proceed. In rare occasions, the agent also crashed to end the episode early. The reward shaping technique helped reduce these failures by influencing the agent to move towards the goal, as observable from the significantly lower failure rates in Table 5. Moreover, these failures from the reward shaping agent were only caused by the agent staying still, which means that the agent with reward shaping is safer to be implemented in real-world environments. Performances of the agents illustrate the impact of the reward shaping technique on improving the agents for simple parking tasks.

IV. CONCLUSIONS

This paper proposed an autonomous parking system using PPO with information from the game-based simulation. Visual information from the front and the rear of the car was applied along with the observed states of the car as the input,

in which the information of states was provided by the simulation. Two PPO agents were developed, trained, and tested in the simulation powered by the Unity game engine. Both agents were trained in the car parking tasks, with differences in the rewarding policies. The first agent was trained with basic rewards provided by the simulation. The second agent was trained with basic rewards and additional rewards through the reward shaping technique. Additional rewards were added to the second agent according to the state of the car after each action. The agents were tested in experiments that required the car to be moved to the parking target from random starting locations 100 times in the simple simulation environment. During the experiments, both agents were able to move the car to the goal successfully with satisfying results. The experimental results dictated that performances of the agent with reward shaping were superior to the first agent with basic rewards in every aspect.

Future work can include the training of the parking agents in more complex scenarios, such as parking in larger areas with more obstacles, parking in a parking building, and parking backwards to specific locations. The trained PPO agents can also be employed in real cars or toy cars for further experiments in real-world environments.

REFERENCES

- [1] S. C. K. Tekouabou, E. A. A. Alaoui, W. Cherif and H. Silkan, "Improving parking availability prediction in smart cities with IoT and ensemble-based model" *Journal of King Saud University - Computer and Information Sciences* 34, 2022.
- [2] G. Ali, T. Ali, M. Irfan, U. Draz, M. Sohail, A. Glowacz, M. Sulowicz, R. Mielnik, Z. B. Faheem and C. Martis, "IoT Based Smart Parking System Using Deep Long Short Memory Network," *Electronics*, no. 1696, 2020.
- [3] K. Korosec, "Daimler and Bosch's driverless parking gets OK to operate without human supervision" TechCrunch Blog, Accessed: Jan. 6, 2026. [Online]. Available: <https://techcrunch.com/2019/07/23/daimler-and-boschs-driverless-parking-feature-can-legally-operate-without-human-supervision/>
- [4] W. Kim and I. Jung, "Smart Parking Lot Based on Edge Cluster Computing for Full Self-Driving Vehicles," in *IEEE Access*, vol. 10, pp. 115271-115281, 2022, doi: 10.1109/ACCESS.2022.3208356.
- [5] A. Uzzaman, M. I. Adam, S. Alan and P. Basak, "Review on the Safety and Sustainability of Autonomous Vehicles: Challenges and Future Directions" *Control Systems and Optimization Letters*, Vol. 3, No 1, 2025.
- [6] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin and Q. -L. Han, "Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses," in *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 7897-7912, Dec. 2021, doi: 10.1109/TII.2021.3071405.
- [7] L. Liu et al., "Computing Systems for Autonomous Driving: State of the Art and Challenges," in *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469-6486, 15 April 2021, doi: 10.1109/JIOT.2020.3043716.
- [8] B. R. Kiran et al., "Deep Reinforcement Learning for Autonomous Driving: A Survey," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909-4926, June 2022, doi: 10.1109/ITITS.2021.3054625.
- [9] M. Toromanoff, E. Wirbel and F. Moutarde, "End-to-End Model-Free Reinforcement Learning for Urban Driving Using Implicit Affordances," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 7151-7160, doi: 10.1109/CVPR42600.2020.00718.
- [10] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang and D. Cao, "Decision-Making Strategy on Highway for Autonomous Vehicles Using Deep Reinforcement Learning," in *IEEE Access*, vol. 8, pp. 177804-177814, 2020, doi: 10.1109/ACCESS.2020.3022755.
- [11] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo and K. Li, "Centralized Cooperation for Connected and Automated Vehicles at Intersections by Proximal Policy Optimization," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12597-12608, Nov. 2020, doi: 10.1109/TVT.2020.3026111.
- [12] E. Beeching, J. Debangoye, O. Simonin, O. Wolf, "Godot reinforcement learning agents". *arXiv preprint*, Dec. 2021 arXiv:2112.03636.
- [13] A. Singh, P. Arora and S. Raheja, "Autonomous Driving System Using CNN and PPO," 2025 3rd International Conference on Advancement in Computation & Computer Technologies (InCACCT), Gharuan, India, 2025, pp. 473-477, doi: 10.1109/InCACCT65424.2025.11011424.
- [14] C. Xu, R. Zhu and D. Yang, "Karting racing: A revisit to PPO and SAC algorithm," 2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI), Kunming, China, 2021, pp. 310-316, doi: 10.1109/CISAI54367.2021.00066.
- [15] Unity Technologies, "Training Configuration File." docs.unity3d.com, Accessed: Jan. 6, 2026. [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.ml-agents@4.0/manual/Training-Configuration-File.html>