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.

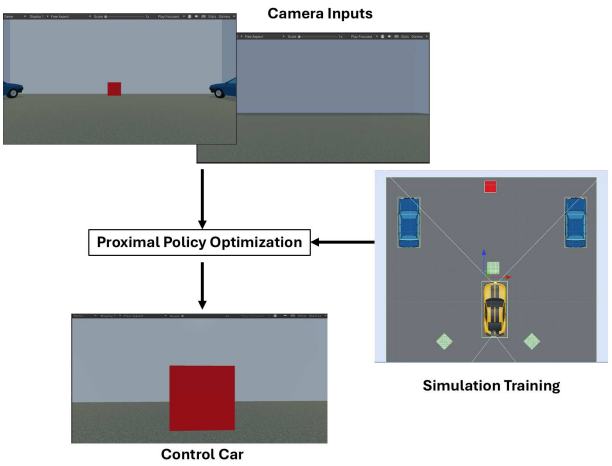


Fig. 1. Diagram of the game-based autonomous parking system.

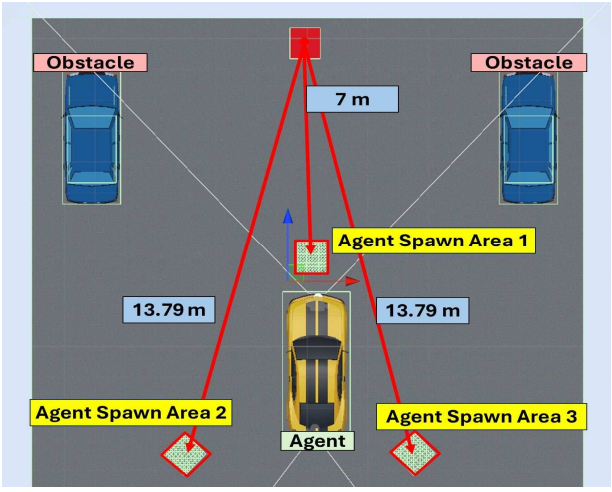


Fig. 2. Top view of the simulation environment.

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 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 I. CAR PARKING SIMULATION SETTINGS

Simulation Properties	Applied Settings
Time Limit per Episode	5 seconds
Camera Inputs	84 × 84 × 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	$1 - (\lvert image\_x_{Goal} - image\_x_{center} \rvert \times 2)$

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.

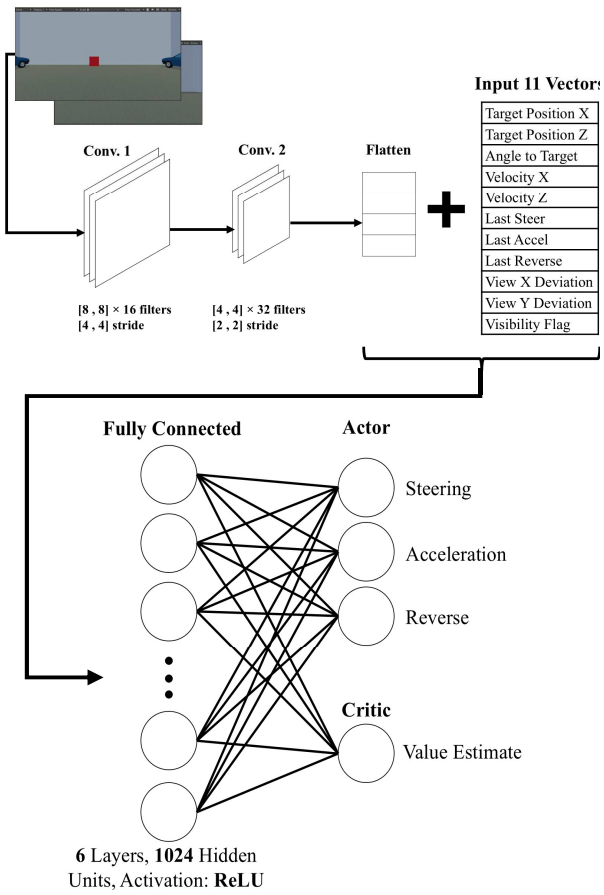


Fig. 3. Structure of the PPO for autonomous car parking agents.

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
lambd	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 "lambd" 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.

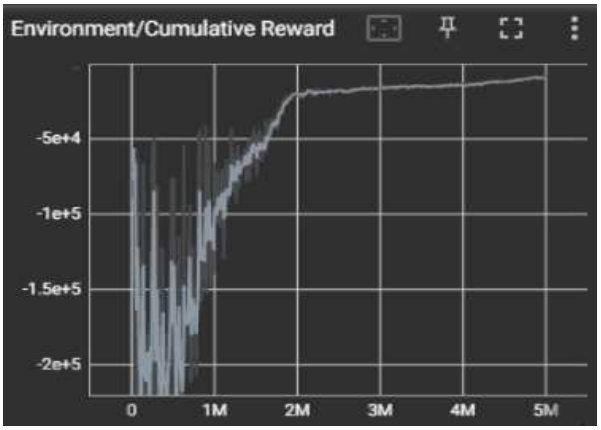


Fig. 4. Cumulative training rewards from the agent with basic rewards.

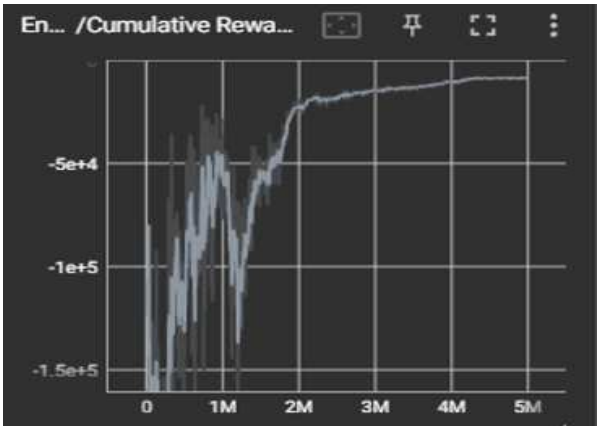


Fig. 5. Cumulative training rewards from the agent with reward shaping.

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

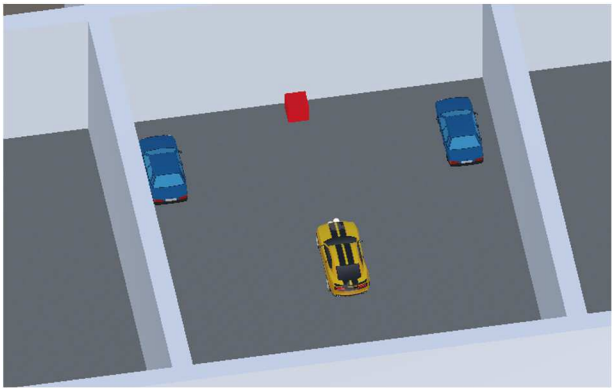


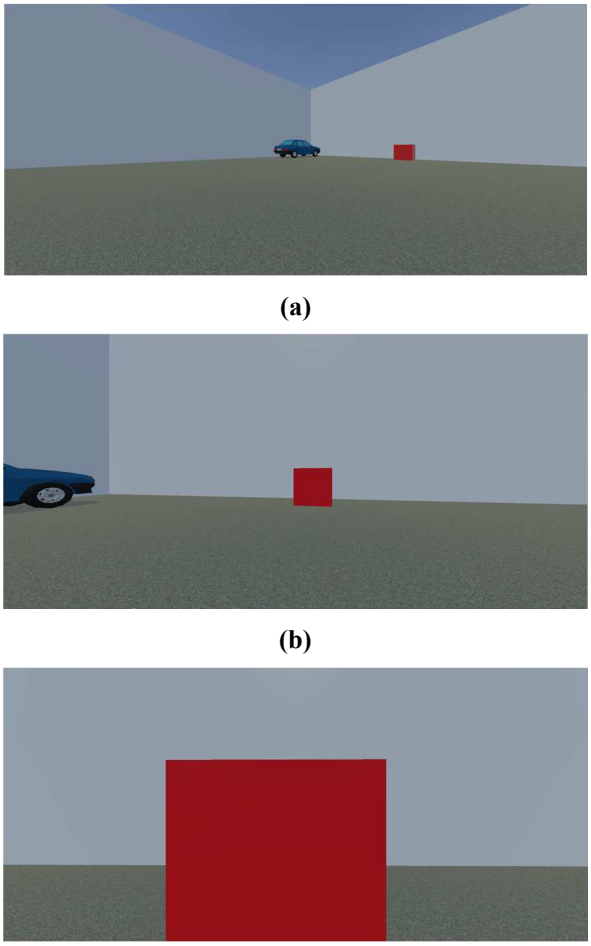
Fig. 6. Example of the agent initial states in car parking experiments.

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 April15, 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] 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