College of Engineering- Center for Environmental Research & Technology

End-to-End Vision-Based Adaptive Cruise Control (ACC) Using Deep Reinforcement Learning

Zhensong Wei (zwei030@ucr.edu), Yu Jiang (yjian091@ucr.edu), Xishun Liao (xliao016@ucr.edu), Xuewei Qi (qixuewei@gmail.com), Ziran Wang (zwang050@ucr.edu), Guoyuan Wu (gywu@cert.ucr.edu), Peng Hao (haop@cert.ucr.edu), and Matthew Barth (barth@ece.ucr.edu)

Introduction

This paper presented a deep reinforcement learning method named Double Deep Q-networks to design an end-to-end vision-based adaptive cruise control (ACC) system. A simulation environment of a highway scene was set up in Unity, which is a game engine that provided both physical models of vehicles and feature data for training and testing. Well-designed reward functions associated with the following distance and throttle/brake force were implemented in the reinforcement learning model for both internal combustion engine (ICE) vehicles and electric vehicles (EV) to perform adaptive cruise control. The gap statistics and total energy consumption are evaluated for different vehicle types to explore the relationship between reward functions and powertrain characteristics. Compared with the traditional radar-based ACC systems or human-in-the-loop simulation, the proposed vision-based ACC system can generate either a better gap regulated trajectory or a smoother speed trajectory depending on the preset reward function. The proposed system can be well adaptive to different speed trajectories of the preceding vehicle and operated in real-time.

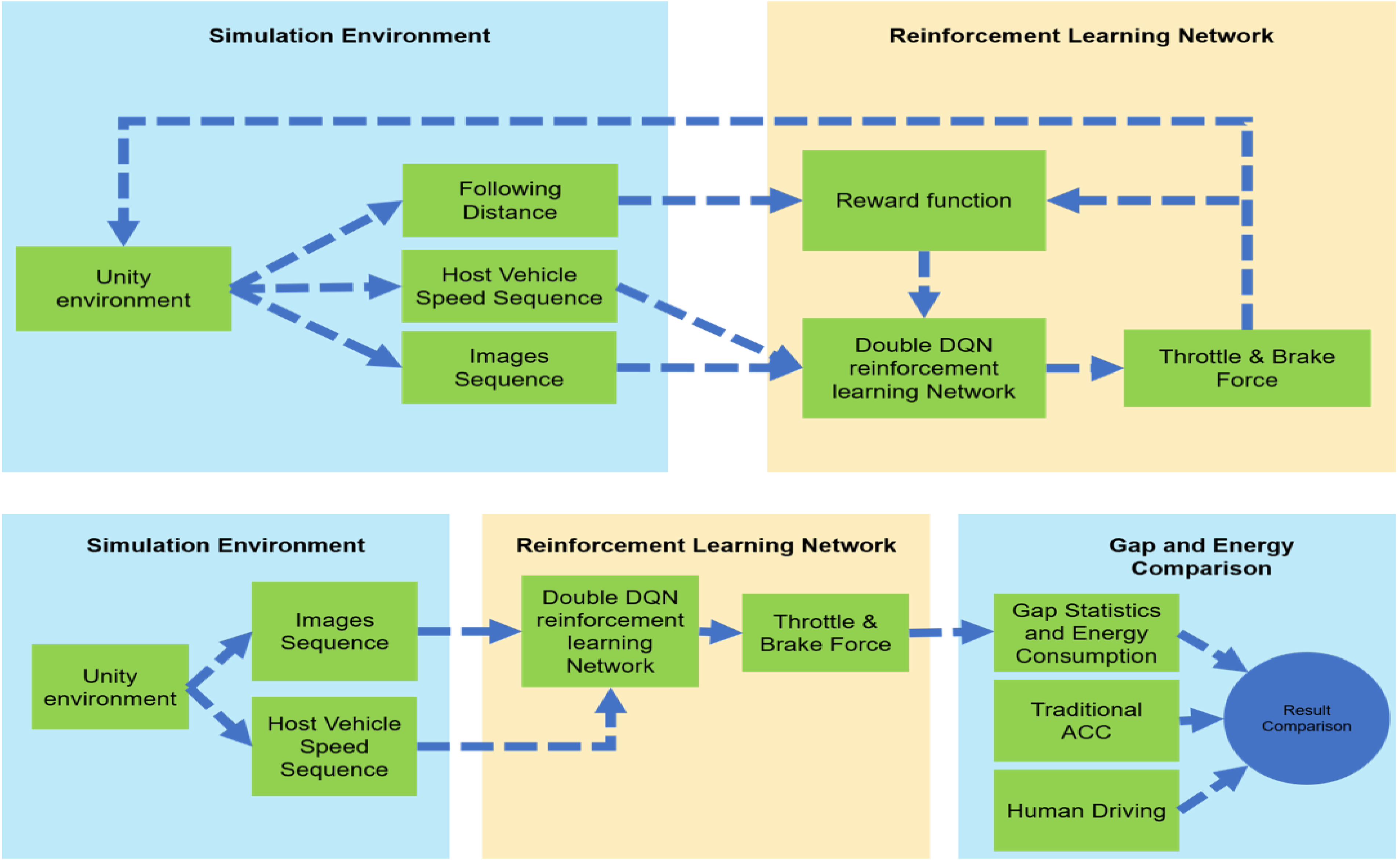


Figure 1. System workflow for training (upper) and testing (lower).

Methodology

- **System Workflow**
  - ✓ System consists of two stages, training and testing. The interaction between the simulation environment and RL network occurs at every time step (see Figure 1).
- **Experiment Setup**
  - ✓ Freeway scene with trees, buildings, and traffic flows are built in the simulation environment (Figure 2).
- **DDQN and Network Structure**
  - ✓ Different reward functions (see Table 1) associated with following distance and throttle/brake force were implemented in the reinforcement learning model for both internal combustion engine (ICE) vehicles and electric vehicles (EV).
  - ✓ The CNN architecture (see Figure 3) applied in the learning receives both image and speed inputs.

Experiment Setup

- **Information Communication**
  - ✓ The information, such as velocity, following distance, and images, is transmitted to the reinforcement learning (RL) network by a socket API using User Datagram Protocol (UDP). The suggested throttle/brake force is calculated and transmitted back to the simulation environment using the same API.
- **Host Vehicle Control**
  - ✓ The host vehicle can be controlled by the suggested throttle/brake force from the RL network.
- **Preceding Vehicle Control**
  - ✓ The velocity data of the preceding vehicle is generated from a large pool of trajectories. Both virtual trajectories and real-world data are used during training and testing.
- **Simulation Environment Reset**
  - ✓ When a collision occurs or the vehicle following distance is greater than 300m, the current episode is terminated, and the simulation environment is reset with all the object settings to be new initial states for the next episode.

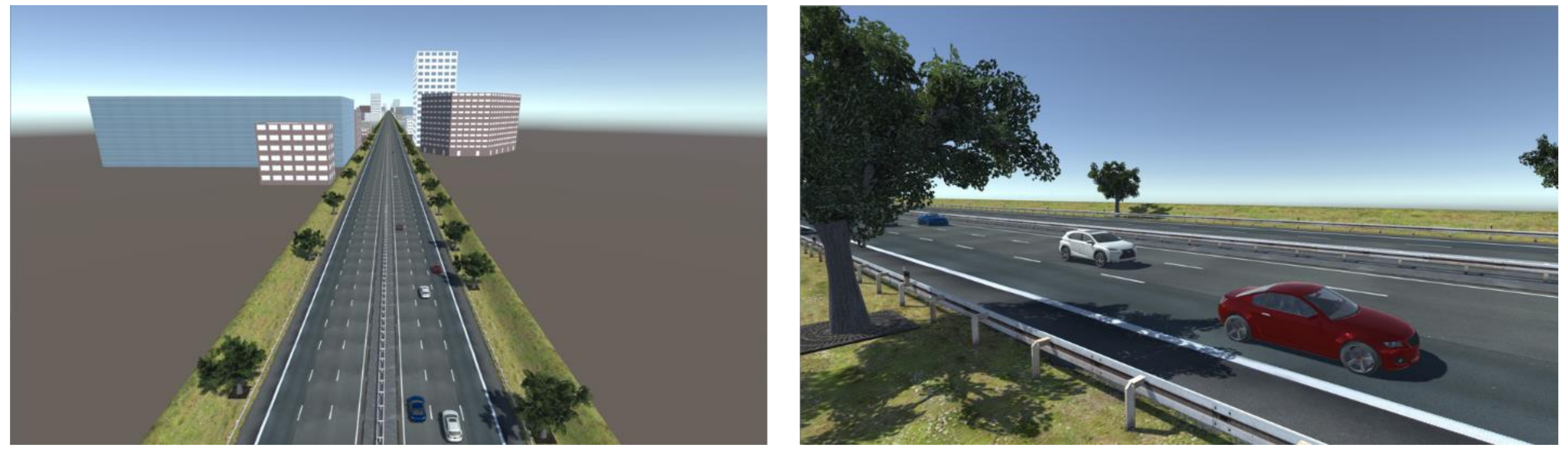
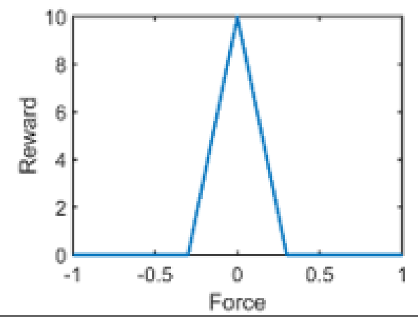
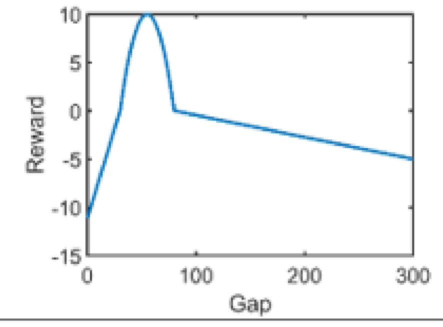


Figure 2. Bird's eye view (left) and elevation view (right) of the simulation environment.

Table 1. Reward Function for Gap-based and Force-based DDQN Model.

Reward Type	Gap-based DDQN	Force-based DDQN
Throttle/Brake Force	None	
Gap		

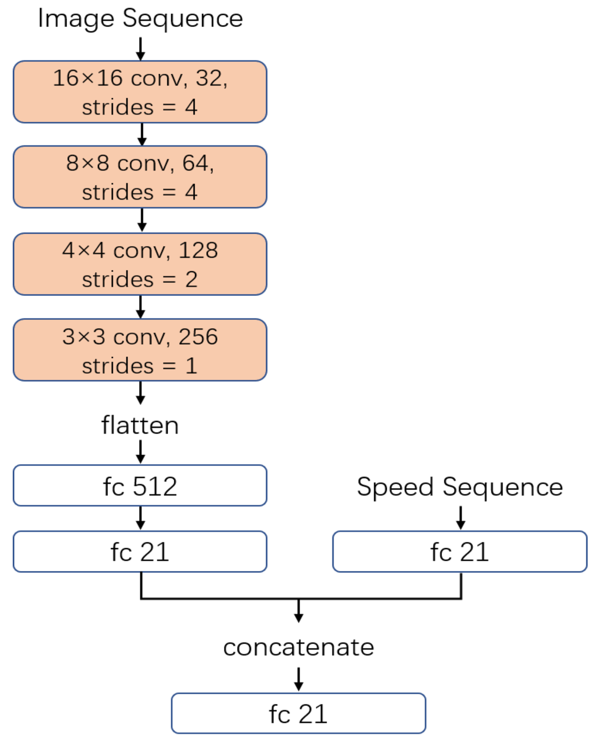


Figure 3. CNN structure in the system

In the network, the image input is a time series sequence of 8 images with size 105 by 150 in grayscale, and the speed input is a time series sequence of 8 speed values of the host vehicle. Conv represents the convolutional layer and fc is the fully connected layer. The 21 output values indicate the 21 Q values of the discretized brake/throttle forces to the vehicle at the given state. We test two different reward functions for ICE vehicles and EVs, one is only related to the gap (gap-based DDQN), while the other one is also related to force (force-based DDQN). We encourage a gap between 30 to 80m and forces from -0.3 to 0.3. The sum of the reward is normalized between -1 to 1 to ensure faster learning.

Case Study

- Simulation Environment: a two-way freeway segment is created in Unity for the simulation study.
- Both virtual trajectories and real-world data are used during training and testing (see Table 2).
- Four trajectories with 4 minutes each are generated to compare between different methods.
- Four driving strategies: 1) gap-based DDQN; 2) force-based DDQN; 3) traditional ACC; 4) human-in-the-loop.

Table 2. Trajectory parameters for ICE Vehicles and EVs in the simulation.

Parameters	ICE Vehicle	Electric Vehicle
PV velocity range (m/s)	[27, 33]	[0, 30]
PV acceleration range (m/s^2)	[-3.5, 3.5]	[-5.5, 3.5]
Initial PV velocity range (m/s)	[27, 33]	[11.2, 15.6]
Initial HV velocity range (m/s)	[25, 30]	[11, 16]
Initial following distance range (m)	[25, 35]	[25, 35]

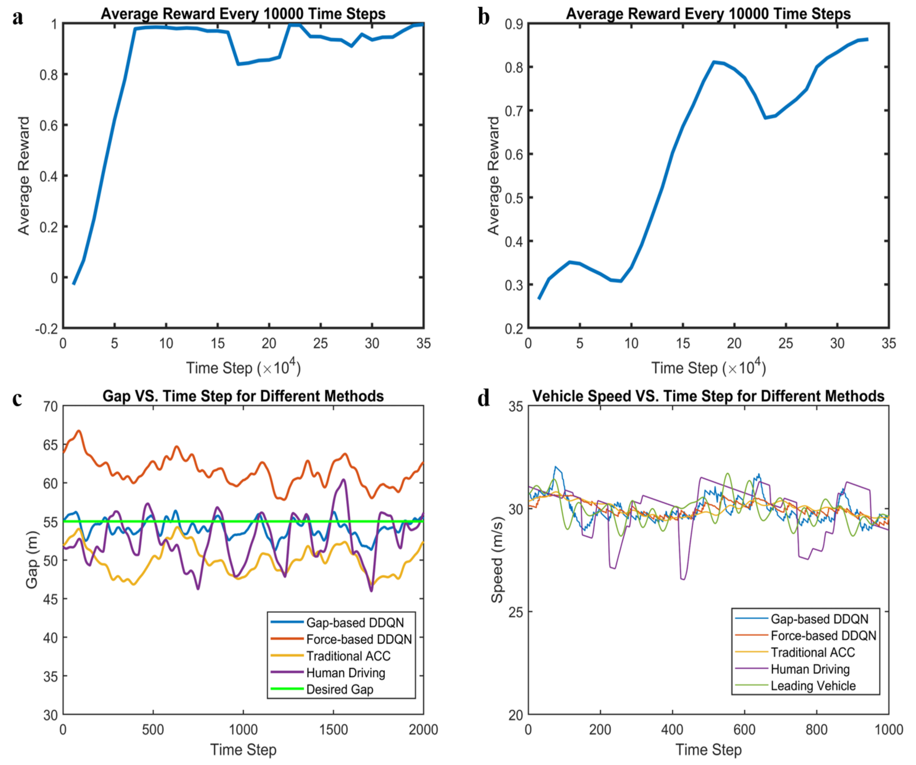


Figure 4. Training and testing result for virtual trajectory.

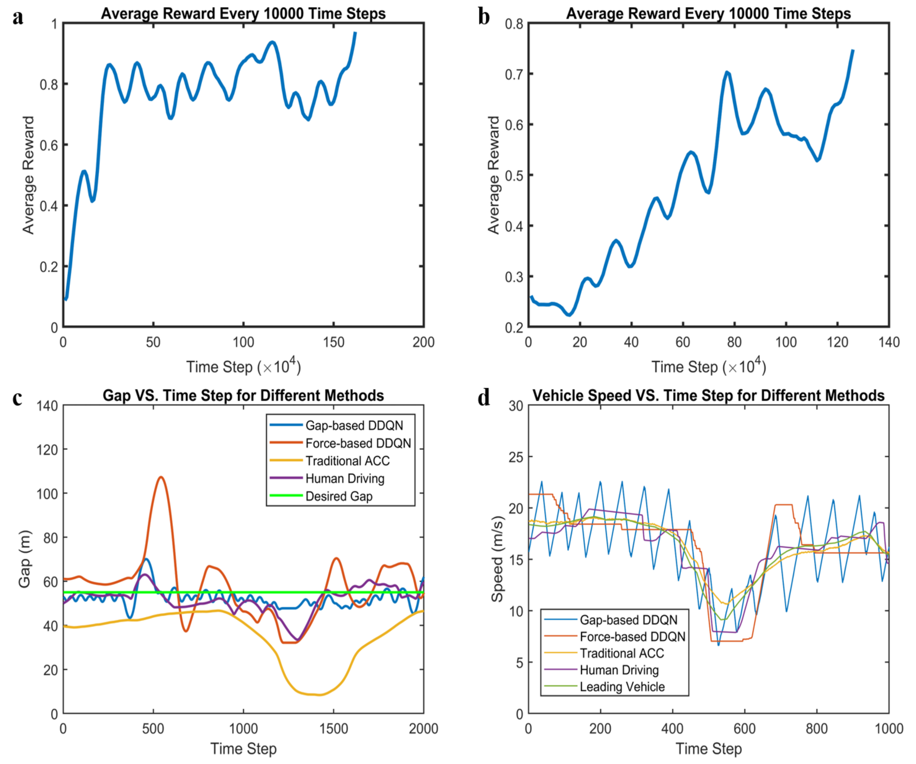


Figure 5. Training and testing result for real trajectory

Table 3. Energy consumption and pollutant emission for combustion engine vehicles between different methods

Method	Leading Vehicle	Gap-based DDQN	Force-based DDQN	Traditional ACC	Human Driving
Fuel Rate (g/mi)	195.393	169.275	143.754	143.805	198.586
CO_2 (g/mi)	346.352	339.069	330.681	337.124	368.524
CO (g/mi)	143.214	99.003	58.119	55.433	143.391
HC (g/mi)	13.613	12.913	9.843	9.443	10.673
NO_x (g/mi)	4.656	3.159	1.841	1.743	4.500
Gap RMSE (m)	N/A	3.423	7.686	6.667	6.591

Table 4. Energy Consumption for EV between different methods

Method	Leading Vehicle	Gap-based DDQN	Force-based DDQN	Traditional ACC	Human Driving
Energy Rate (KJ/mile)	1.135e3	4.914e3	2.056e3	1.128e3	1.500e3
Gap RMSE (m)	N/A	5.083	10.98	17.440	5.439

Conclusions and Future Work

- An end-to-end vision-based ACC using DRL under Unity simulation environment was proposed.
- Training and testing result over virtually generated and real-world driving trajectories show the effectiveness and robustness of the following ability for both ICE or EV models.
- The inference time of 3.58 ms indicates the real-time working ability of the proposed method.
- Further research will be conducted to integrate the energy model into DRL reward function and apply other reinforcement learning algorithms and recurrent neural network (RNN) structures, such as Actor-Critic and Long short-term memory (LSTM).