

Road Tracking Using Deep Reinforcement Learning for Driving Cars Applications^{*}

Raid Rafi Omar Al-Nima^{1,2}, Tingting Han², and Taolue Chen²

¹ DCSIS, Birkbeck University of London

² Technical Engineering College of Mosul, Northern Technical University, Iraq

Abstract. Deep reinforcement learning has recently aroused wide attentions. It combines deep learning with reinforcement learning to solve difficult tasks. This paper proposes an efficient deep reinforcement learning network to solve road tracking problems. The network collects input states from forward car facing views and produces suitable road tracking actions. The actions are derived from encoding the tracking directions and movements. We explored both value iteration and policy iteration based deep reinforcement learning algorithms. A comparison between the two and a comparison with existing networks are provided. Our results are very promising - the best driving accuracy achieved 93.94%, the highest in the comparison.

Keywords: Road Tracking, Deep Reinforcement Learning, Markov Decision Process

1 Introduction

Road tracking is one of the most challenging tasks emerged from key applications such as autonomous driving. It aims to automatically guide a car through the correct track without crashing other cars or objects. Reinforcement learning (RL) is about an agent interacting with the environment, learning an optimal policy, by trial and error, for sequential decision making problems. The combination of deep neural networks and reinforcement learning gives rise to deep reinforcement learning (deep RL). We hypothesise that deep reinforcement learning can be one of the most promising methods to address the road tracking issue and this paper reports our approaches and results to apply DRL to accomplish some typical tasks in road tracking. One of the primary tasks in road tracking is to track objects [1–4]. Other tracking tasks include tracking and controlling velocity [5], tracking periodic gene repressilator [6], tracking single and double pendulum [7], and tracking maximum powers for wind speed conversion systems [8]. Applying DRL techniques to road tracking has been reported, but has not been explored thoroughly. Perot *et al.* investigated tracking roads of a driving car. A deep reinforcement learning was used [9]. The main problem in this work is that the driving car oscillates around the main road track. This is due to the selected

^{*} This work is supported by EPSRC grants EP/P00430X/1 and EP/P015387/1.

reward in this study, where it utilized the oriented angle of the road with the car speed. Yun *et al.* [10, 11] proposed an action-decision network (ADNet) to track objects, where a pre-trained Convolutional Neural Network (CNN) was firstly employed then the reinforcement learning was utilized. Our work contributes to this area by suggesting an effective road tracking procedure and employing the MDP based on the deep reinforcement learning, where states are used as inputs; rewards are utilized to evaluate the tracking policy and actions are predicted to provide new states. Effective coding for various road tracking possibilities of actions has been considered such as turning to the left or right and recognizing crossing object(s).

Related work. For tracking objects, Grigore and Grigore [1] proposed a tracking system controller by using a recurrent neural network with the reinforcement learning. Cohen and Pavlovic [2] suggested an effective and vigorous real-time tracker, which utilized RL and was applied to tracking personal faces. In 2004, Liu and Su proposed an object tracking method by exploiting the reinforcement learning. The reinforcement learning was employed here to determine the features of the tracking object [3]. In 2017, Supančič and Ramanan constructed a learning policy for tracking objects. The reinforcement learning was applied to video streams to provide on-line decision [4].

2 Modelling

In this section, we are to address various essential issues to model the road tracking, such as how to identify road crossing object(s), how to encode different tracking directions, and how to design the deep reinforcement learning network to determine the next action, etc.




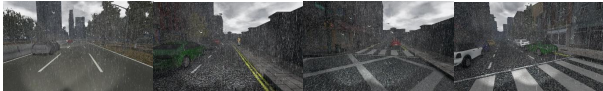
The database. Our research is based on the SYNTHIA-SEQS-05 database [12]. Video sequences were recorded and saved, from which one extracted front, rear, left and right view road images around the car for both right and left steering cars. Each view covers a range of angle up to 100 degrees. As a result, a large number of simulated image frames were provided, each has a resolution of $760 \times 1,280 \times 3$ pixel.

The images were taken in four different environments: (1) clear environment in spring, (2) fog, (3) rain and (4) heavy-rain environment. In our study, we look at *forward* view road frames for a *right* steering car in all *four* environments. Examples from the four databases are given in Table 1. In our study, we look at *forward* view road frames for a *right* steering car in all *four* environments.

2.1 Road tracking

To differentiate the main objects on the road, the databases provide specific colours for certain objects in the images, e.g., sky is grey, buildings are brown,

Table 1: Examples of the four employed environments

Environment	Examples
Spring	
Fog	
Rain	
Heavy-rain	

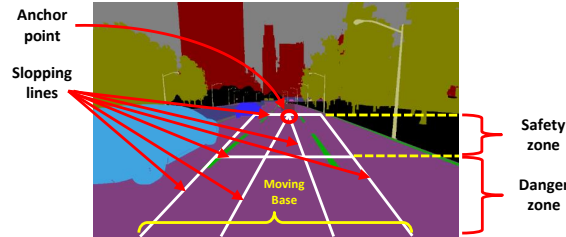


Fig. 1: The suggested front view road tracking zones, lines and anchor point

roads are purple, side walks are blue and road markings are green. Overall, the purple and green colours refer to the allowed driving regions.

Based on the images, we further divide each view (image) into a safety zone and a danger zone. If objects are spotted in the safety zone the car can keep moving, as the zone is further away, allowing ample time to stop or slow down the car if necessary. The danger zone is the area that a car must stop if any objects are recognised. Such zones can be found in Fig. 1.

To determine a safety zone or a danger zone, a virtual triangle and trapezoid are drawn. The base of the triangle and trapezoid moves in the road direction, and the crossing lines slope toward the tracking direction. The height of the triangle (and the trapezoid) is two thirds of the image. The top one third of the trapezoid is the safety zone and the bottom two thirds is the danger zone. The shared point of the triangle and the trapezoid is called an *anchor point*. It denotes the road tracking direction by following the track centre.

By drawing the virtual triangle in the trapezoid area we can divide the region into left, right and centre region, with which the direction of any crossing objects

can be specified. The car can, for instance, avoid an object crossing from the left side by moving to the right, if the space is empty there. If objects are identified from both sides in the danger zone, then the car has to stop.

2.2 Actions codes

At any point, a car can take one of the following actions: drive straight on, turn left or right, reverse and stop. In our work, we use a five-digit binary number to encode each action, see the table below.

Action sign	Binary code	Equivalent decimal code	Description
Straightforward	01110	14	follow the straight track
Turn left	11100	28	follow the track to the left
Turn right	00111	7	follow the track to the right
Stop	00000	0	object(s) identified from both sides
Backwards	11111	-31	reverse

When a crossing object is detected, it will change the code from that side with a “0”. For instance, a car is moving forward with action 01110, then an object appears in the danger zone from left, the action code becomes 00111, indicating that the car turns right to avoid the object. **TT: You didn’t say when to turn 0 to 1. For instance why 01110 becomes 00111? It is only stated when 1 should be turned to 0 (with object appearing in the danger zone).** :TT If another object appears from the right, the code will be 00110 and the car will have to stop. A car has to stop as long as there are no three consecutive 1’s. We model all such cases as 00000 to reduce the number of coding values. Fig. 2 shows various coding cases. The cases in the red dashed box are all coded as 00000. The binary codes can be converted to desired equivalent decimal codes in the standard way. For instance, $(01110)_2 = (14)_{10}$ and $(00111)_2 = (7)_{10}$. The only exception here is the backward direction, where a negative sign (-) is added to refer to the reverse movement. It is worth mentioning that the reverse action will only be considered if the car is being out of the track. The decimal codes are used in the regression layer of the proposed deep reinforcement learning network as will be explained later.

2.3 Proposed DRL-RT

In this section, we describe the neural network (NN) used in the Deep Reinforcement Learning framework for Road Tracking (DRL-RT), which consists of eight layers: two convention layers, two Rectified Linear Unit (ReLU) layers, a pooling layer, a fully connected layer, a regression layer and a classification layer. Fig. 3 depicts the proposed DRL-RT network.

The input of the NN is an image of a car facing view, and it is considered as the current state. The dimension of the input image is reduced to $254 \times 427 \times 3$ pixels to speed up the training. The first layer is a convolution layer which

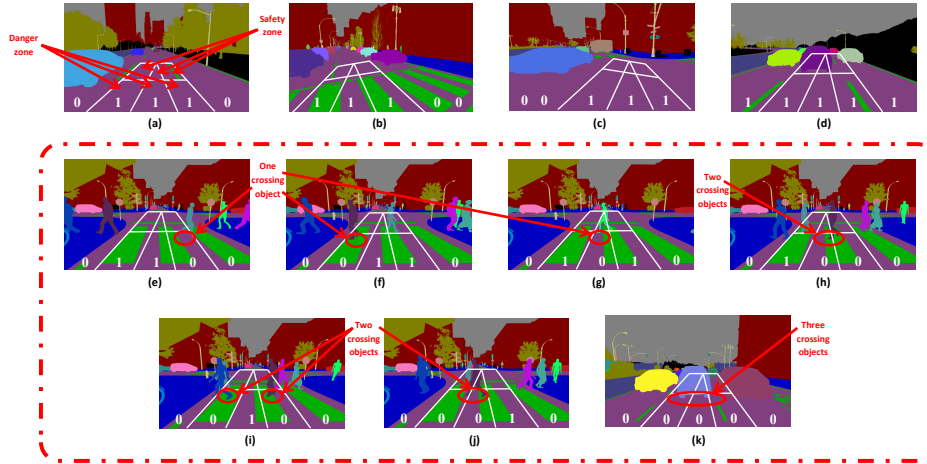


Fig. 2: Segmented images for road tracking: (a) straightforward, (b) turning left, (c) turning right, (d) reverse or backward, (e-g) stopping action because of a single crossing object, (h-j) stopping action because of two crossing objects and (k) stopping action because of three crossing objects

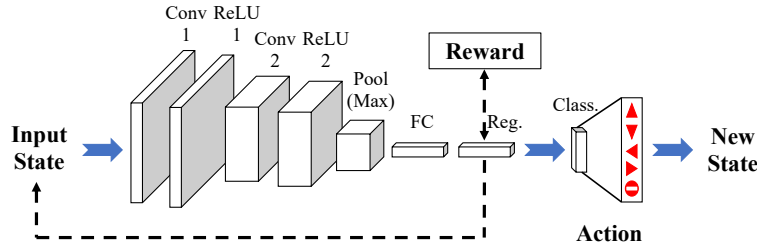


Fig. 3: The main design of the proposed DRL-RT. It consists of two conventional layers, two ReLU layers, a pooling layer, a fully connected layer, a regression layer and a classification layer

consists of 5 filters, each of which has a filter size of 10×10 pixels. This layer is to extract the main features of the input image. A ReLU layer is used next and it removes the negative values and maintain the positive values of the previous layer. The 3rd layer is also a convolution layer, consisting of 5 filters each of which has a filter size of 5×5 pixels. It extracts more features from the input images. A ReLU layer is employed again in the 4th layer. This layer rectifies the negative values. It has empirically been established that using two convolution layers with two ReLU layers can well analyse the information before being compressed by applying the next layer. Then, a pooling layer of a maximum type is applied as the 5th layer the filter size here is 3×3 pixels with a stride of 3 pixels.

TL: the sixth layer is to encode the policy? :TL The 6th layer is a fully connected layer. It collects the outputs from the previous layer and produces a series of decimal code tracking values. **TT: The decimal code is generated here in the 6th layer and then in the 7th layer? :TT** In the 7th regression layer, a series of directional road tracking codes are generated. **TL: what is successful unsuccessful code? :TL** The successful code in this layer produce positive rewards, whereas, unsuccessful codes generate negative rewards. The network should propagate the information forward and backward to update the network weights, during

the training stage till obtaining as many positive rewards as possible. Given the codes in the regression layer, it is the classification layer’s task to generate a new action - one of the five as in Section 2.2.

We are to address a few technical details in the DRL-RT network. The underlying model of the network is a Markov Decision Process (MDP).

- *States*. The states in the MDP are the views (images).
- *Rewards*. The reward R takes a simple form, i.e., the correct tracking **TT: what do you mean by correct tracking? How to tell whether a tracking is correct? Comparing to the training set? Elaborate here. :TT** is considered as (+1), whereas, the reward R of the incorrect tracking is considered as (-1). Measuring the successful process of the road tracking will be based on obtaining as many positive rewards as possible.
- *Policy search*. The DRL-RT network is based on the policy search. DRL-RT collects input images as current states S_t (or current views) and gets advantages from rewards R to generate actions A . The actions then predict new states S_{t+1} by following the track of the road. The process will be repeated in a multi-episodic manner.

3 Results

3.1 Implementation

All implementations were performed on a PC with 8 GB RAM and Intel Core i5 processor (3.2 GHz). Only the microprocessor was used in this study to train and test the DRL-RT. We used 270, 284, 268 and 248 frames for the environments of spring, fog, rain and heavy-rain, respectively. The frames are equally divided between the training and testing stages (50% each), where the odd-indexed frames are used in the training stage and the even-indexed frames are used in the testing stage.

3.2 Training stage

The suggested DRL-RT network has been separately trained for each environment. The following parameters have been assigned for the trainings: Adaptive Moment Estimation (ADAM) optimizer [13], learning rate equal to 0.0003, gradient decay factor (β_1) equal to 0.9, squared gradient decay factor (β_2) equal to 0.99 and mini batch size equal to 128.

The training performance of the DRL-RT for the four databases is given in Fig.4. This figure demonstrates the relationships between the Root Mean Square Error (RMSE) and the training iterations during the NN training stages. The RMSE values are usually exploited to demonstrate the differences between desired values and output values. These differences are usually reduced during the proceeding of training iterations. Clearly, the curves are successfully declined toward goals.

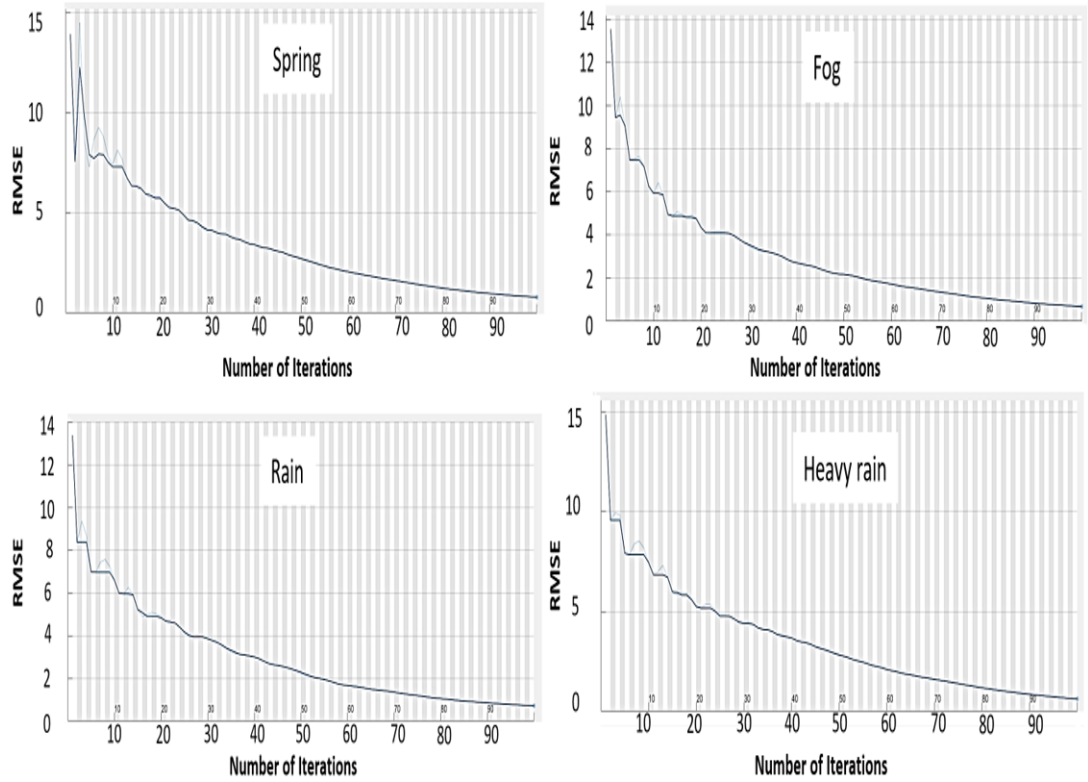


Fig. 4: The training RMSE vs #iterations for the four environments

3.3 Testing stage

In the testing stage we use (driving) accuracy to indicate how well our model performed. **TT: Define driving accuracy. :TT** Fig. 5 shows that the driving accuracy attained its highest value of 93.94% by using the spring environment database. This is because that the DRL-RT has analysed very clear provided images. The fog environment database obtained a high driving accuracy of 93.66%. Here, the overall views are blurred but the road tracking can still be distinguished **TT: What is the road tracking in this case? Why can they still be distinguished in foggy environment? :TT**, and the accuracy is only slightly lower than the spring views. The driving accuracy of the rain environment database achieved 89.55% and this is due to the noise effects of rain drops on image views. Finally, the inferior driving accuracy of 84.68% was recorded for the heavy-rain environment database as the amount of rain drops (or noise) is increased here.

3.4 Comparisons

We have investigated and simulated various deep learning approaches to establish a fair comparisons between our DRL-RT method and other networks. Table 2

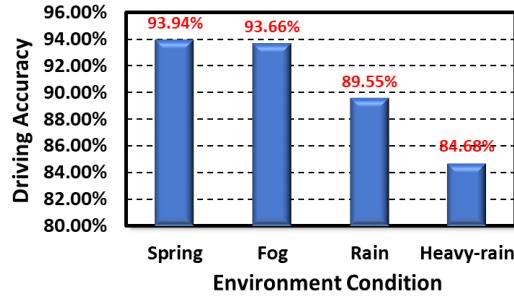


Fig. 5: The accuracy of the DRL-RT under different environments

shows the accuracies of different deep learning networks by applying the database of SYNTHIA-SEQS-05-SPRING, some parameters were reasonably changed to allow acceptable comparisons. The reason of selecting this database here is that it has the clear environment, which is suitable to discard undesired effects and provide fair judgement.

Table 2: A comparison between the DRL-RT method and other suggested networks

Reference	Deep learning method	Accuracy
Karaduman and Eren [14]	CNN	67.42%
Bojarski <i>et al.</i> [15]	CNN	74.24%
George and Routray [16]	CNN	83.33%
Yun <i>et al.</i> [10, 11]	ADNet	83.33%
Mnih <i>et al.</i> [17]	DQN	88.64%
Proposed method	DRL-RT	93.94%

From Table 2, it can be seen that the suggested CNN in [14] obtained the inferior tracking accuracy of 67.42%. This is due to the architecture of this network, where it constructed to classify directions of traffic signs. More specifically, a pooling layer was applied after each convolution layer and no ReLU layers were used. This caused compressing and wasting useful extracted features after each convolution layer.

The CNN used in [15] attained a low accuracy of 74.24%. The main drawback of this network is that it considers steering angles to be tracked, where this increases the erroneous of obtaining precise outputs.

The CNN used in [16] achieved 83.33%. This is also due to the architecture of this network, which was designed for classifying eye gaze directions.

The ADNet used in [10, 11] attained the same accuracy of 83.33%. The essential problem here is represented by the considered rewards, which were basically designed for recognizing moved objects as the rewards are updated in the stop

action. In addition, the Adnet architecture is not so appropriate for road tracking tasks.

The Deep Q-Network (DQN) used in [17] and illustrated in [18] achieved reasonable accuracy of 88.64%. This can be due to the DQN processes, where it combines between the deep network and the Q-learning. This network can be considered as the nearest method to our approach.

Finally, our proposed method has shown superior performance by attaining the accuracy of 93.94%. This can be due to the overall structure of our road tracking method including the network architecture, tracking policy and designed codes.

4 Conclusion

A deep reinforcement neural network is established for road tracking termed the DRL-RT. This network is suggested to guide driving cars under different weather environments. Different tracking instances were coded to represent the appropriate road tracking. The MDP concept is used here, where the network accepted states and produced actions by taking advantages from rewards. The proposed deep reinforcement network is based on the policy search. This study were compared with other work and it reported superior performance. In addition, interesting results were benchmarked by the proposed approach. That is, the best performance of 93.94% was recorded for a clear environment and reasonable performance were reported under unclear environments of fog, rain and heavy-rain as the obtained accuracies were respectively equal to 93.66%, 89.55% and 84.68%.

References

1. O. Grigore and O. Grigore, "Reinforcement learning neural network used in a tracking system controller," in *Proceedings 9th IEEE International Workshop on Robot and Human Interactive Communication. IEEE RO-MAN 2000 (Cat. No.00TH8499)*, pp. 69–73, 2000.
2. A. Cohen and V. Pavlovic, "Reinforcement learning for robust and efficient real-world tracking," in *20th International Conference on Pattern Recognition*, pp. 2989–2992, 2010.
3. F. Liu and J. Su, "Reinforcement learning-based feature learning for object tracking," in *Proceedings of the 17th International Conference on Pattern Recognition. ICPR 2004.*, vol. 2, pp. 748–751, 2004.
4. J. Supančič and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 322–331, 2017.
5. X. Jinlin, Z. Weigong, and G. Zongyang, "Neurofuzzy velocity tracking control with reinforcement learning," in *9th International Conference on Electronic Measurement Instruments*, pp. 3–465–3–468, 2009.
6. A. Sootla, N. Strelkowa, D. Ernst, M. Barahona, and G. Stan, "On periodic reference tracking using batch-mode reinforcement learning with application to gene

- regulatory network control,” in *52nd IEEE Conference on Decision and Control*, pp. 4086–4091, 2013.
7. J. Hall, C. E. Rasmussen, and J. Maciejowski, “Reinforcement learning with reference tracking control in continuous state spaces,” in *50th IEEE Conference on Decision and Control and European Control Conference*, pp. 6019–6024, 2011.
 8. C. Wei, Z. Zhang, W. Qiao, and L. Qu, “Reinforcement-learning-based intelligent maximum power point tracking control for wind energy conversion systems,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 10, pp. 6360–6370, 2015.
 9. E. Perot, M. Jaritz, M. Toromanoff, and R. d. Charette, “End-to-end driving in a realistic racing game with deep reinforcement learning,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 474–475, 2017.
 10. S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, “Action-decision networks for visual tracking with deep reinforcement learning,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1349–1358, 2017.
 11. S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, “Action-driven visual object tracking with deep reinforcement learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2239–2252, 2018.
 12. G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, 2016.
 13. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
 14. M. Karaduman and H. Eren, “Deep learning based traffic direction sign detection and determining driving style,” in *International Conference on Computer Science and Engineering (UBMK)*, pp. 1046–1050, 2017.
 15. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.
 16. A. George and A. Routray, “Real-time eye gaze direction classification using convolutional neural network,” in *International Conference on Signal Processing and Communications (SPCOM)*, pp. 1–5, 2016.
 17. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
 18. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.