Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

Article

Intelligent Land Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning

Lingli Yu [1,2,3], Xuanya Shao [1,*] Yadong Wei [1] and Kaijun Zhou [4]

[1] School of Information Science and Engineering, Central South University, Changsha, China; llyu@csu.edu.cn
[2] State Key Laboratory of Robotics and System, Harbin Institute of Technology, Haerbin, China
[3] State Key Laboratory of Mechanical Transmissions, Chongqing University, Chongqing, China
[4] School of Computer and Information Engineering, Hunan University of Commerce, Changsha, China; alpha218@126.com
* Correspondence: 174611083@csu.edu.cn; Tel.: +86-130-5516-6724

Abstract: Aiming at the problem of model error and tracking dependence in the process of intelligent vehicle motion planning, an intelligent vehicle model transfer trajectory planning method based on deep reinforcement learning is proposed, which obtain an effective control action sequence directly. Firstly, an abstract model of the real environment is extracted. On this basis, Deep Deterministic Policy Gradient (DDPG) and vehicle dynamic model are adopted to jointly train a reinforcement learning model, and to decide the optimal intelligent driving maneuver. Secondly, the actual scene is transferred to equivalent virtual abstract scene by transfer model, furthermore, the control action and trajectory sequences are calculated according to trained deep reinforcement learning model. Thirdly, the optimal trajectory sequence is selected according to evaluation function in the real environment. Finally, the results demonstrate that the proposed method can deal with the problem of intelligent vehicle trajectory planning for continuous input and continuous output. The model transfer method improves the model generalization performance. Compared with the traditional trajectory planning, the proposed method output continuous rotation angle control sequence, meanwhile, the lateral control error is also reduced.

Keywords: intelligent driving vehicle; trajectory planning; end-to-end; deep reinforcement learning; model transfer

1. Introduction

Although intelligent driving technology is developing rapidly, some new problems are emerging during the development. In 2016, the first major accident of Tesla was happened in the field of automatic driving. Meanwhile, Uber occurred an accident of automation driving hitting pedestrians on March 28, 2018. These problems greatly arouse worldwide attention on the safety of intelligent driving. Therefore, it is still a long way for intelligent driving to improve its innovative and stable safety. As the key of its technology, trajectory planning technology is attracting more and more attention and exploration by researchers at home and abroad.

Trajectory planning is not only applied to intelligent vehicles, but also widely used in the field of robotics and unmanned aerial vehicles[1,2]. There are various ways of trajectory generation in trajectory planning, including Nelson polynomial, spiral curve equation, spline curve, Bezier curve etc [3]. For example, 4th-order polynomial and dynamic bicycle model are utilized to describe vehicles kinematics model [4], considering the overtaking and chasing behavior of different cost functions in each case. However, it assumes that the vehicle velocity is a constant, which conflicts with most actual situations. Ref.[5] puts forward a technique of trajectory smoothing and stitching based on Bezier Curve. Ref.[6] proposes a trajectory planning based on Bezier Curve that takes the command of

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

kinematics constraint, initial state constraint, target state constraint, and curvature continuous constraint into consideration.

With the rapid development of deep learning, vision-based control methods acquire great achievements [7]. Hotz [8] adopts VAE and GAN to achieve image coding, road tracking and intelligent driving vehicle potential space decoding. Ref.[9] low-level control strategy and advanced prior action are learned through neural network, and multi-level strategies are taken as heuristic search algorithms to realize complex motion planning tasks. In Ref.[10], deep learning models are adopted to establish the mapping relationship between lidar distance, target position and control instruction. To realize the motion planning of intelligent vehicle, Ref.[11,12] propose deep learning to establish the mapping relationship between control sequence and corresponding trajectory.

In recent years, reinforcement learning has been applied to robot control tasks. In Ref.[13], Deep Q Network (DQN) is proposed to deal with discrete action continuous state, which goals at combining deep neural network with reinforcement learning. Subsequently, Ref.[14,15] offer an off-line depth reinforcement learning algorithm based on deep Q network and extend it to continuous high-dimensional state space. Ref.[16,17] make it possible to achieve multiple targets by extending DQN. Ref.[18] proposes prior experience replay technology to improve the performance of DQN. Ref.[19] exposit experience to improve sample collection efficiency. OU process [20] adds noise after action strategy to improve network exploration ability. Ref.[21] introduces network parameter hierarchy with noise to improve network performance. Ref.[22] shows the possibility of learning complex manipulation strategies without demonstrations. Ref.[23] adopts deep reinforcement learning model to establish traffic signal agent. While Ref.[24] solve the problem of complex traffic intersection without traffic signal. Ref.[25] propose a motion planning method without map. The sparse sensor ranging information and target position are utilized as input, and the continuous steering command is taken as output, and verification is conducted in practical experiments. In Ref.[26], data efficiency and task performance are improved by addressing the problem of maximizing cumulative rewards for reinforcement learning(RL), and considering supervised/unsupervised learning styles, so as to achieve navigational capabilities.

As we all known that Q-Learning solve the low-dimensional problem of discrete space. DQN improves the processing ability of high-dimensional state space, but it is still difficult to cope with high-dimensional continuous action space. The Actor-Critic method handle with continuous action space, but randomness strategy makes it difficult for the network to converge. To this end, Deep Deterministic Policy Gradient (DDPG) adopts the Actor-Critic framework to combine the advantages of DQN to solve the problem of continuous state space and continuous action space. Meanwhile, it adopts a deterministic policy to ensure the network more convergent. Since traditional motion cannot eliminate the model error, an end-to-end model transfer trajectory planning based on depth reinforcement learning is proposed. Furthermore, DDPG is a deep reinforcement learning method, it is utilized to train the model in a simple virtual environment which is constructed independently, meanwhile, their dependence is reduced on the sample data. Meanwhile, it can deal with the model training of continuous input and continuous output, thus directly outputs the control action and trajectory sequence. The complexity is lower than that of the optimal control calculation, and the model transfer method is applied to improve the model generalization performance. Compared with traditional planning and end-to-end planning, the proposed method has more continuous corner control sequence and smaller lateral error while the vehicle is driving.

2. Reinforcement Learning and Description of Driving Environment

2.1. Reinforcement Learning Method

The basic principle of reinforcement learning is shown in Figure 1. When the agent is required to achieve a task, it first interacts with environment (***Env***) via action(***a***), then the impact of action on the environment brings agent into a new state(***s***). At the same time, agent receives reward feedback (***Reward***) from the environment. Agent and environment generate a large amount of data through continuous loop and interaction. The reinforcement learning utilizes these sample data to adjust the

strategy π. Afterwards it interacts with ***Env*** to enter new ***state*** and generate new $data = (s_t, a_t, r_t, s_{t+1})$. After that, the new samples are adopted to modify the strategy π for several times. After a great deal of iterative learning, the agent finally learns the optimal strategy π^* to complete the corresponding task.

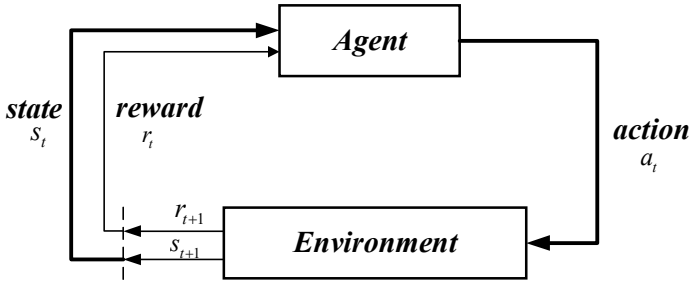


Figure 1. The principle of reinforcement learning

Strategy π refers to the agent select action ***a*** under state ***s***, it is the key problem in reinforcement learning. The strategy π is a map from the agent aware of environmental state ***s*** to action ***a***. The random strategy selects corresponding action according to the probability $\pi(a \mid s)$ of each action; Deterministic policy selects actions $a = \pi(s)$ directly according to ***s***.

$$\begin{aligned} &stochastic\ Policy: \quad \sum \pi(a \mid s) = 1 \\ &deterministic\ Policy: \ \pi(s): S \to A \end{aligned} \tag{1}$$

Calculating cumulative rewards or returns when a strategy π is given. The definition of cumulative returns:

$$G_t = R_{t+1} + \gamma R_{t+2} + \mathrm{L} \ = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2}$$

Here, $0 < \gamma < 1$ is the discount factor of long-term income, meanwhile, state function is defined as the cumulative return benefit corresponding to state ***s*** under ***a*** strategy π:

$$\upsilon_\pi(s) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \tag{3}$$

The corresponding state-action value function is defined as:

$$q_\pi(s,a) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \tag{4}$$

2.2. Virtual Driving Environment Model Design

According to the description of intelligent vehicle driving scene, intelligent driving behavior decision tasks include normal driving, changing overtaking, curve/ramp driving and so on. Here, if environment model is built as a circular map with three lanes in Figure 2. Specifically, the cyan area and outer lane boundary is seemed as insurmountable obstacles, while the others are free travel space. The light blue lines indicate the desired path with rewards $path_d = (X_d, Y_d, \phi_d)$. Intelligent vehicle $Car = (x_c, y_c, \varphi_c, v)$ drives in a circular map and learns intelligent driving maneuver, including straight, changing, and curving driving behavior. Where x_c, y_c, φ_c represents current position and posture information of the intelligent vehicle.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

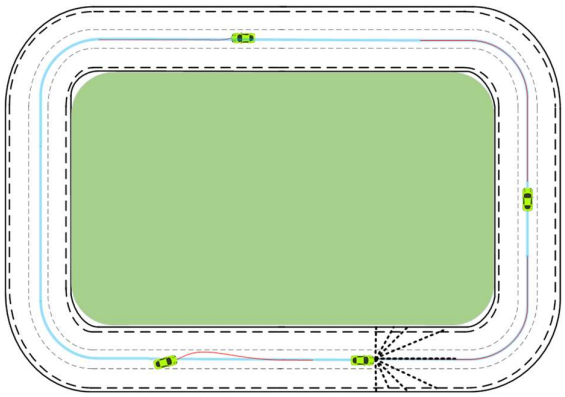


Figure 2. Virtual environment abstract model

To simplify the environment model ***Env***, the intelligent vehicle has ***n*** ranging beams ***Sensor.*** And the farthest distance d_{Max} of each ranging beam is the same. What's more, each ranging beam support feedback information $Sensor_i = (d_i, x_{end}, y_{end})$ to intelligent vehicle when it encounters obstacles. Here, d_i is the length that ranging beam is blocked by obstacles or boundaries, x_{end}, y_{end} is position coordinate of the beam in contact with obstacles or boundaries. The intelligent vehicle speed keeps a constant v, and angle control output is $-\delta_{\min} \le \delta_t \le \delta_{\max}$. Thus, the key return function ***Reward*** for the environment model ***Env*** is:

$$Reward = \begin{cases} -1 & when\ terminates \\ R_{action} + R_{money} & else \end{cases} \tag{5}$$

$$\begin{cases} R_{action} = -\lambda_1 * \left\| \delta_{old} - \delta \right\|^2 \\ R_{money} = \begin{cases} 0 & get\_money = False \\ 0.1 & get\_money = True \end{cases} \end{cases} \tag{6}$$

$$get\_money = \begin{cases} True & if\ |\Delta x| \le \varepsilon_1\ \&\ |\Delta y| \le \varepsilon_2\ \&\ |\Delta \varphi| \le \varepsilon_3 \\ False & other\ else \end{cases} \tag{7}$$

where, λ_1 is the positive penalty coefficient, R_{action} represents the difference penalty between the front and rear successive front wheel angles δ_{old}, δ of an intelligent vehicle. And the smaller change between successive actions, the smaller penalty. R_{money} represents the reward for a intelligent vehicle driving on the desired path, $(\Delta x, \Delta y, \Delta \varphi)$ is the difference between current posture (x_c, y_c, φ_c) and desired path $path_d = (X_d, Y_d, \phi_d)$, $\varepsilon_1, \varepsilon_2, \varepsilon_3$ is the fault-tolerant error.

Intelligent vehicles randomize initial position (x_0, y_0, φ_0) according to the given policies π_{reset} to ensure a more adequate exploration of the environment and the stability of result. Intelligent vehicle termination condition includes, 1) contact with obstacle or boundary; 2) meet the maximum number of driving steps $n_{step} = Num_{\max}$. The optimal intelligent driving maneuver for intelligent vehicle learning is π, therefore, the strategy space of model is $\pi_{all} = \{\pi_{reset}, \pi\}$.

$$\pi_{reset} : \begin{cases} x_0 \in [X_{\min}, X_{\max}] \\ y_0 \in [Y_{\min}, Y_{\max}] \\ \varphi_0 \in [\phi_{\min}, \phi_{\max}] \end{cases} \tag{8}$$

State space is assumed as $\sum(Sensor) = \{d_0, d_1, ..., d_n\}$, and motion space is assumed as $\sum(\delta) = \{\delta_{new}\}$. For intelligent vehicle and environment model ***Env***, the abstract model ***M*** is constructed.

$$M = \{Env, Car, \sum(Sensor), \sum(\delta), \pi_{all}, Reward\}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

3. Model Transfer Trajectory Planning Based on Deep Reinforcement Learning, DRL-MTTP

3.1. DDPG Network Structure and Algorithm Flow

For complex continuous state space $\sum(Sensor)$ and continuous action space $\sum(\delta)$, it is significant to train deep reinforcement learning model M_θ in virtual environment $\boldsymbol{M}$ by DDPG algorithm. DDPG is consisted of Actor policy network and Critic evaluation network in Figure 3. The state $s_{sensor} \in \sum(Sensor)$, speed v and the action δ_{old} of the last moment are combined as $s_a = (s_{sensor}, v, \delta_{old})$. They are adopted as the input of Actor policy network. Meanwhile, the policy network hidden layer utilizes three full connected networks. Each layer contains 512 neurons. Full connection is followed by BN(batch normalization), then Relu is adopted as the activation function. At the same time, the last layer of the network chooses *tanh* as the activation function to map the network output between the interval [-1,1]. The network output is action $\delta \in \sum(\delta)$. After state s_a and action δ is merged as $s_c = (s_a, \delta)$, then it is input to Critic evaluation network. The hidden layer of evaluation network and the policy network hold the same structure, but its last layer is activated by linear function such as E.q.(9). Thus, the network output is the corresponding Q value $Q(s_a, \delta)$ of s_c.

$$y = kx + b \tag{9}$$

where x is the input of last layer, y is the predicted Q value, k, b is the weight and bias for network training.

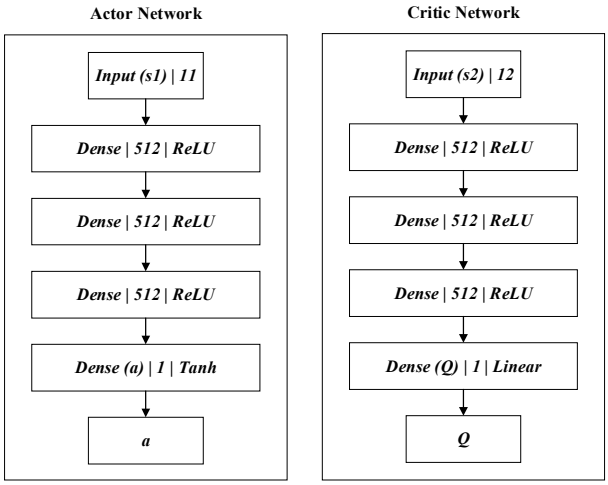


Figure 3. DDPG network structure

DDPG adopts Actor-Critic framework, including Actor and Critic structure. Here, the Actor part includes the online policy network and the target policy network, the deterministic policy is adopted to get a definite action from current state. The Critic part includes Online Q network and Target Q network, Bellman equation of the action-state function Q are utilized to measure the quality of action. The pseudo code of the DDPG algorithm is shown in Table 1, and the DDPG algorithm flow is shown in Figure 4, input state $(d_1, ..., d_9, v, \delta_{old})$ and output action δ are represented respectively. DDPG adopts a deterministic policy, and the policy output is an action. Therefore, it needs less sampled data, meanwhile, the algorithm keeps more efficient. However, the disadvantage is that the environment cannot be explored. In order to improve the algorithm exploration ability, OU stochastic process is added to the deterministic policy action. Furthermore, environmental execution is carried out after sampling from random process of the action. Due to its good correlation in time series, OU stochastic processes are enable to explore environments with momentum properties by the agents.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

Table 1. Pseudo code of DDPG algorithm

Pseudo code of DDPG algorithm
1. Randomly initialize Critic Online Q network parameters θ^Q and Actor's Online policy network parameters θ^μ.
2. Initialize Critic Target Q network parameters $\theta^{Q'} \leftarrow \theta^Q$ and Actor's Target policy network parameters $\theta^{\mu'} \leftarrow \theta^\mu$.
3. Initialize Experience Replay Memory（**R**）.
4. **for** *episode* = 1, M **do**
5. Initialize the OU random process *D* for the exploration of action.
6. Input initial observation state s_1.
7. **for** $t = 1,\ T$ **do**
8. Choose action a_t based on current strategy $\mu(s_t)$ and exploring noise D_t: $a_t = \mu(s_t) + D_t$.
9. Perform the action a_t, get the reward r_t, and observe the new state s_{t+1}.
10. Store the process (s_t, a_t, r_t, s_{t+1}) in ***R***.
11. Sampling from R to get the process (s_i, a_i, r_i, s_{i+1}) of batch ***N***.
12. Set $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$// Q' is the state-action value calculated by the Target Q network, and μ' is the current strategy obtained by the Target policy network.
13. Update Critic's Online Q network by minimizing the loss function: $L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^Q\right)^2\right)$.
14. Update the Actor's Online Policy Network with Sampling Gradient: $\nabla_{\theta^\mu}\mu \mid s_i \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a \mid \theta^Q\right)\mid_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s \mid \theta^\mu\right)\mid_{s_i}$.
15. Update Critic's Target Q Network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$.
16. Update Actor's Target Policy Network: $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$.
17. **end for**
18. **end for**

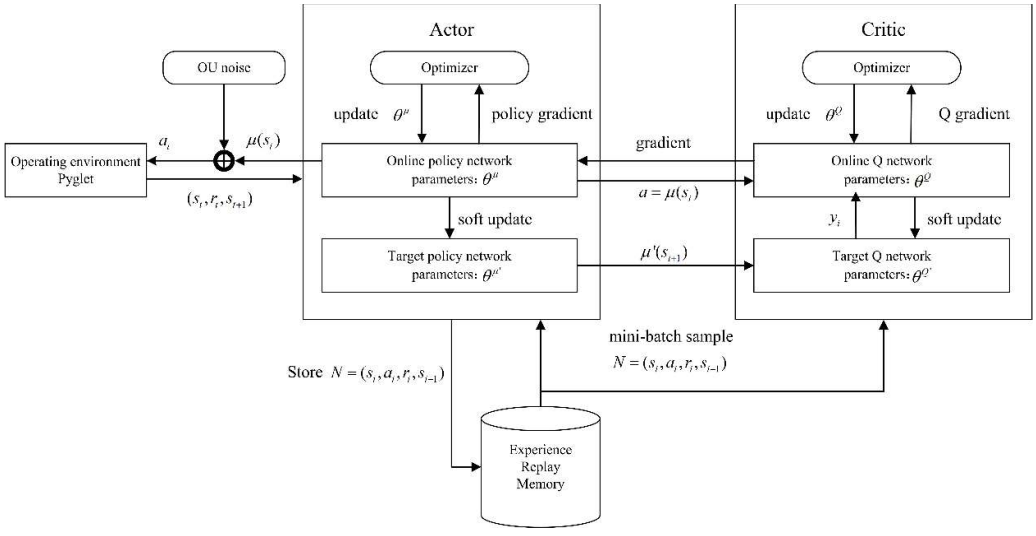


Figure 4. DDPG algorithm flow chart

3.2. Model Transfer Strategy

The intelligent vehicle ***Car*** obtains optimal intelligent driving strategy π from environment model ***M*** through deep reinforcement learning training. Besides, the real environment ***M**** is obviously different from the virtual environment model ***M***, the former usually tends to be more complex and time-varying. Therefore, if the training model of virtual environment ***M*** is directly applied to real environment ***M****, it brings numerous predictable or unpredictable problems.

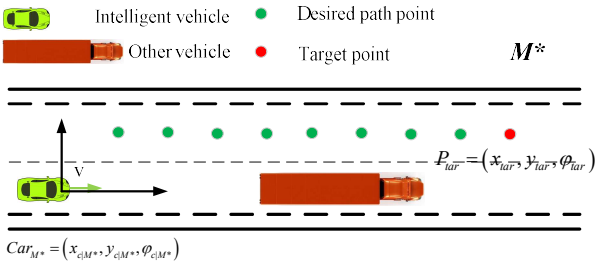
The decision tasks of intelligent vehicles include straight, changing lanes, crossing corners and ramps driving, etc. So the model of intelligent driving tasks is abstracted from real environment ***M**** and migrated to virtual environment ***M***, which is mapping into location area corresponding to ring map in ***M***. Then the optimal driving strategy π is adopted to plan control-trajectory sequence $C=\{\delta,\zeta\}$ that achieve driving task, including control sequence $\delta=\{\delta_1,\delta_2,...,\delta_t\}$ and its corresponding trajectory $\zeta=\{p_1,p_2,...,p_t\}$. Finally, intelligent vehicles are carried out task δ to complete the driving task in real environment ***M****.

According to different sub-task ϕ, such as lane keeping, lane changing and overtaking, the fixed reference points $P_{ref}|\phi=(x_{ref},y_{ref},\varphi_{ref})$ in ***M*** is set as a reference target or local tasks. In terms of driving task endpoint goal $P_{tar}=(x_{tar},y_{tar},\varphi_{tar})$ of path planning and intelligent parking posture $Car_{M^*}=(x_{c|M^*},y_{c|M^*},\varphi_{c|M^*})$ in real environment ***M****, the model transfer strategy $\Re$ is mapped to ***M*** by Eq.(10). Finally, the intelligent parking posture is obtained as $Car=(x_c,y_c,\varphi_c)$.

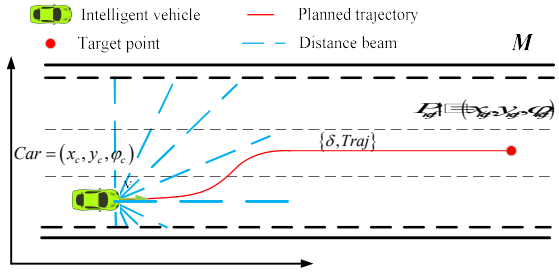
$$\Re:\begin{cases}\theta=\varphi_{tar}-\varphi_{ref}\\(x',y',\varphi')=(x_{c|M^*}\cos\theta-y_{c|M^*}\sin\theta,x_{c|M^*}\sin\theta+y_{c|M^*}\cos\theta,\varphi_{c|M^*}-\theta)\\(x_{tar}',y_{tar}',\varphi_{tar}')=(x_{tar}\cos\theta-y_{tar}\sin\theta,x_{tar}\sin\theta+y_{tar}\cos\theta,\varphi_{ref})\\(\Delta x,\Delta y,\Delta\varphi)=(x'-x_{tar}',y'-y_{tar}',\theta)\\(x_c,y_c,\varphi_c)=(x_{ref}+\Delta x,y_{ref}+\Delta y,\varphi_{ref}+\Delta\varphi)\end{cases}\tag{10}$$

where θ is the difference between target heading angle φ_{tar} of vehicle driving destination and heading angle φ_{tar} of reference point. In order to keep the heading angle of target point P_{tar} in real environment coincide with heading angle of reference point P_{ref} in virtual environment, the real environment coordinate system is rotated by θ. (x',y',φ') is the pose corresponding to ego vehicle in rotated coordinate system, and $(x_{tar}',y_{tar}',\varphi_{tar}')$ is the pose corresponding to the target point ***P*** in the rotated coordinate system. $(\Delta x,\Delta y,\Delta\varphi)$ is the difference in pose between the ego vehicle and the target point in the rotated coordinate system. And (x_c,y_c,φ_c) is the corresponding pose of the ego vehicle in the virtual environment after the model transfer.

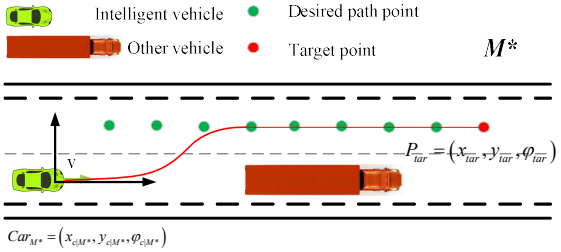
Figure 5(a) shows the real-world ***M**** scene. The road condition information is composed of invariants and variables. The invariants include the number of lanes and width of lanes. The variables are the information of obstacles, which is feedback by the distance beam of intelligent vehicle. The initial planning process is to travel along the current driving lane. When it is detected that there is a vehicle with lower speed than the ego vehicle in front of current lane, the left lane is taken as a desired path. The driving task is to switch to the left lane based on the behavioral decision planning. The green vehicle is the current vehicle position $\mathbf{x}=[x,y,\varphi,v,\omega]^T$ and the pose information is $Car_{M^*}=(x,y,\varphi)$, the green dot in the center of left lane is the scattered point of path planning. Here, the red point is the destination of current driving task $P_{tar}=(x_{tar},y_{tar},\varphi_{tar})$. Figure 5(b) shows the scene after Car_{M^*},P_{tar} is migrated to $\Re$ through the model. Mapping to the virtual environment ***M***, the intelligent vehicle pose is $Car=(x_c,y_c,\varphi_c)$. Acquire $\sum(Sensor)$ according to the distance beam in ***M***, then merge δ_{old} into state s, and the control sequence $\delta=\{\delta_1,\delta_2,...,\delta_t\}$ and the trajectory sequence $\zeta=\{p_1,p_2,...,p_t\}$ are obtained by model M_θ. Figure 5(c) shows the corresponding scene of control sequence $\delta=\{\delta_1,\delta_2,...,\delta_t\}$ and track sequence $\zeta=\{p_1,p_2,...,p_t\}$ in the real environment ***M****.



(a)The scene of real environment M*



(b)Scene after model transfer $\Re$



(c)The corresponding scene of real environment M^*

Figure 5. Transition diagram of the switch task model

3.3. Algorithm Framework of Model Transfer Based on Deep Reinforcement Learning

DRL-MTTP aims to abstract the complex real environment and transfer it to a simple virtual environment through the model. Furthermore, the optimal intelligent driving strategy is applied to the virtual environment, which is trained by the agent after deep reinforcement learning. Thus, the optimal trajectory control sequence is obtained to realize the end-to-end trajectory planning in the real environment. Figure 6 is technical diagram of DRL-MTTP.

The framework DRL-MTTP is shown in Table 2. Firstly, sub-task γ and path planning data sets are initialized according to upper data streams. After that, going into trajectory planning stage. Then an appropriate target point P_{target} is selected as local goal based on the sub-task γ. Thus, the selection process is shown in Equation (11):

$$\begin{cases} s_{t-1,t} = \sqrt{\left(x_t - x_{t-1}\right)^2 + \left(y_t - y_{t-1}\right)^2} \\ l^2 = \sum_{t=1}^{target} s_{t-1,t} \end{cases} \tag{11}$$

where, (x_i, y_i) are the corresponding coordinates of P_i. $s_{t-1,t}$ is the straight-line distance between P_{t-1} and P_t. l is the arc length threshold of path scatter points which is determined by sub-task γ. And P_{target} is the target point that satisfy the threshold requirement. Subsequently, target set is obtained by adding noise to the target point P_{target}, and $(\varepsilon_x, \varepsilon_y, \varepsilon_\varphi)$ satisfies Gaussian distribution. Furthermore, corresponding position Car of intelligent vehicle in the virtual environment M is

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

calculated by model transfer for each target point P_{target}. Meanwhile, the status of environment ***s*** and the status of intelligent vehicle ***x*** are gained by observing its range light beam in M.

During the planning time T, deep reinforcement learning model M_θ is utilized in each unit of time ***t*** to analyze state s and predict the action δ_t. At the same time, the dynamic model is adopted to simulate the prediction action. Meanwhile, environment state s and intelligent vehicle state x are updated, and track ζ_t is recorded. Finally, control-trajectory sequence pair $C=\{\delta,\zeta\}$ under real environment M^* is acquired by model transfer.

The second stage is about optimal trajectory selection. In this stage, the evaluation function of each control-trajectory pair is calculated. At the same time, the collision probability of the trajectory ζ is also judged. The control-trajectory pair with minimum ***J*** and no collision trajectory are decided as the optimal trajectory.

$$\min J = \kappa_1 \int_0^T \left(\Delta\delta^2\right)dt + \kappa_2 \left[h\left(\zeta_T\right) - h\left(\zeta_{target}\right)\right] \tag{12}$$

where, ***T*** is the termination time, $\Delta\delta$ is the difference of continuous action, κ_1,κ_2 is the weight coefficient, and h represents the rectangular area of vehicle outline at the end of vehicle trajectory.

Finally, the result of planning is executed. The intelligent vehicle implements the first τ steps of control sequence δ. And the model reference control is utilized to enhance the robustness and reduce the influence of system error by the model error. If the task is not terminated, planning action of the first two phases is repeated to realize the intelligent vehicle dynamic trajectory planning.

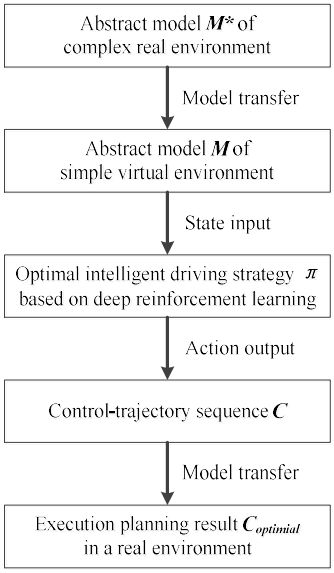


Figure 6. block diagram of DRL-MTTP

Table 2. DRL-MTTP frame

DRL-MTTP frame
1. initialize ***terminal***, $S_1:\{P_1,P_2,...,P_m\}$; //receive tasks and data from the top
2. **while** $terminal = false$
3. //trajectory planning stage
4. $P_{target} \longleftarrow S_1:\{P_1,P_2,...,P_m\}$; //select a target from a path planning scatter set S_1 based on a task
5. **for** $i=0,\ N$ **do**
6. $P_{target,i} = P_{target} + \xi_{noise}$; //add noise to generate target sets
7. $Car_i \xleftarrow{\Re}\left(P_{target},P_{ref},Car_{M^*}\right)$; // calculate the corresponding position and pose of the intelligent vehicle by model transfer
8. $s,\mathbf{x} \xleftarrow{M}\left(Car_i,Sensor\right)$; // calculate state based on ranging light beam

for $t = 0,\ T$ **do**

  $\delta_t \xleftarrow{M_\theta} s$; // calculate action by deep reinforcement learning model

  $\mathbf{x} = \int_0^{\Delta t} f(\mathbf{x}, \delta_i)$; // dynamic model simulation

  $\zeta_i \leftarrow \mathbf{x}$; // record track

end for

$C_i : \{\delta, \zeta^*\} \xleftarrow{\Re} (\delta, Car_i, Car_{M^*}, \zeta)$; //calculate control-trajectory sequence pair by model transfer

end for // optimal trajectory selection stage

$J_{\min} = \infty$;

for $i = 0,\ N$ **do**

  $J = \kappa_1 \int_0^T (\Delta\delta^2) dt + \kappa_2 \left[h(\zeta_T) - h(\zeta_{target}) \right]$; // calculate and obtain evaluation function

  if $J < J_{\min}$ **and** no collision //select the control-trajectory pair with minimum J value and no collision as the optimal trajectory

    $J_{\min} = J$;

    $C_{optimal} = C_i$;

  end if

end for

// stage of execute the planning result

$\delta_{optimal} : \{\delta_1, \delta_2, ,, , \delta_\tau\}, \tau \le T$; //get the results of the first τ steps

update ***terminal***; // whether the task ends or not

end while

4. Simulation Test on Trajectory Planning of Intelligent Vehicle

4.1. Deep Reinforcement Learning Model Training

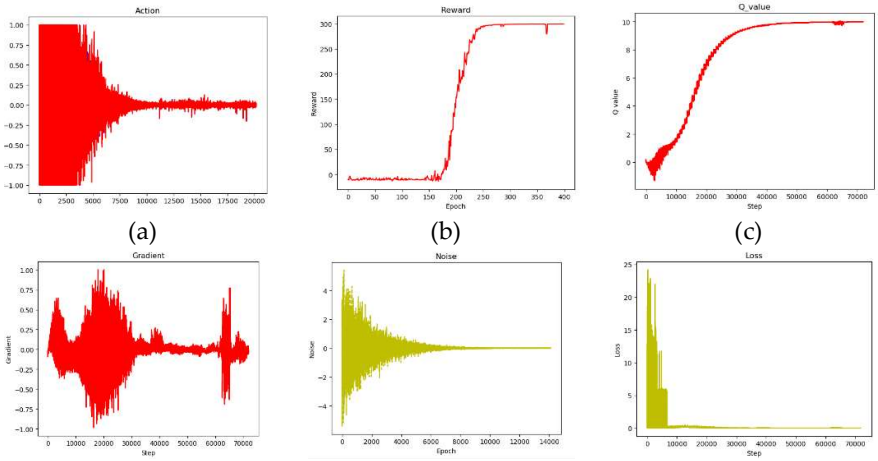
In the virtual simulation environment ***Env***, the circle map is 100m long and 50m wide. The driveway is 3.4m wide, and size of intelligent vehicle is 2m×4m. Here speed $v = 36km/h$, which keeps a constant. The dimension of ranging light beam $n = 9$, and the farthest ranging $d_{Max} = 20m$, the largest output of angle $\delta_{\max} = 0.3rad$. The desired $path_d$ is the center line of middle lane. Besides, errors $\varepsilon_1 = \varepsilon_2 = 0.1m, \varepsilon_3 = 0.5236$, continuous action penalty coefficient λ_1=0.01, and random initial position $x_0 \in [10, 980], y_0 \in [10, 50]$, $\varphi_0 \in [-\pi/2, \pi/2]$. Maximum step number $Num_{\max} = 600$, and step gap is 0.1s. Meanwhile, software environment is a Linux operating system with 16G memory, and its graphics card is GTX1080 Ti. Meanwhile, it takes advantage of the deep learning framework TensorFlow.

With the greater learning rate, the less effect of previous training is retained. Similarly, the greater discount factor, the more emphasis on experience. The smaller discount factor, the more attention is paid to the current return. If the number of hidden layers and hidden layer neurons are too less, and the data are fitted well. Conversely, if the number of hidden layers and hidden layer neurons are too larger, which leads to over-fitting easily. Therefore，a better network structure and network parameters are designed after several trials. The hyperparameters in the deep reinforcement learning model M_θ are set as follows. The discount factor $\gamma = 0.9$, learning rate of the network of Actor and Critic are both 10^{-4}, the optimization method of Adam [27] is adopted, soft update rate τ=0.001, hidden layer neurons number is 512, and the size of experience replay pool is 10^4, the size of batch is 64, the error is generated by Gaussian process. Then, initial variance $var_{\max} = 2$, minimum variance $var_{\min} = 0.01$, and the attenuation rate is 10^{-4}.

As is shown in Figure 7, the rewards and the average Q value of agent gradually increase with the number of iterations during the training process. Finally, it tends to be stable. While the loss gradually decreases to 0 with the increasing of iteration times, it indicates the evaluation of network is more and more effective. As the number of iterations increases, the noise decreases continuously. Meanwhile, the agent with sufficient capability of exploration and detection are provided in the early

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

stage, and the sufficient mining ability are embodied in the later stage. Figure 7 shows that the model learns from experience, and approaches to the optimal strategy continuously.



(a) (b) (c)

(d) (e) (f)

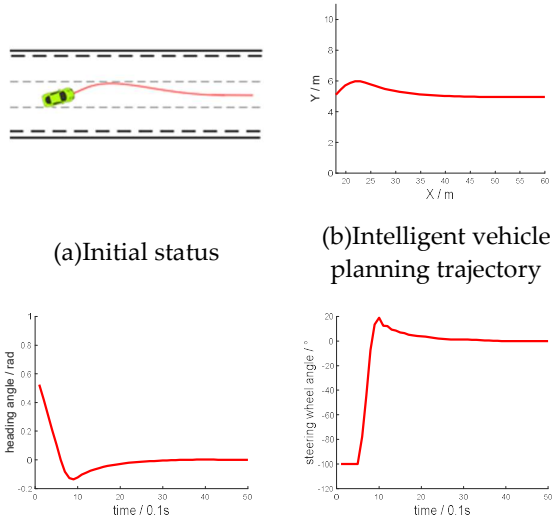
Figure 7. Training results of deep reinforcement learning model

4.2. Verification of Intelligent Vehicle Trajectory Planning Based on DRL-MTTP

4.2.1. Trajectory Planning of Lane Keeping

During the simulation test, the intelligent vehicle speed is set as 10m/s (36km/h). The intelligent vehicle starts from the center position of the middle lane at 30° yaw angle and -30° yaw angle respectively, which is shown in Figure 8(a) and Figure 8(e). The abscissa of Figure 8(b) and Figure 8(f) are "X/m" and their ordinate is "Y/m". The abscissa of Figure 8(c) and 8(g) are "time/0.1s", and their ordinate are "heading angle/rad". Figure 8(d) and 8(h) abscissa are "time/0.1s" , and the ordinate is "steering wheel angle/°".

Through the algorithm model, the steering wheel control sequence is adjusted to keep lane. At beginning, the intelligent vehicle has a deviation in the initial position, and it reaches a stable lane keeping after adjustment. Currently, the front wheel angle average value of three experiments is $-0.0001635rad$ ($-0.009367°$). Therefore, the steering angle is stable around 0°, with the mean value of $-0.1562°$ .Meanwhile, the average lateral deviation after stabilization is 0.04cm.



(a)Initial status

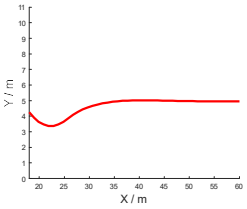
(b)Intelligent vehicle planning trajectory

(c) Heading angle of intelligent vehicle

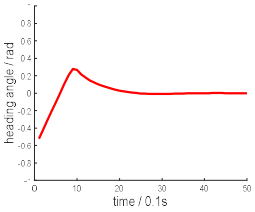
(d)Steering wheel angle control sequence

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

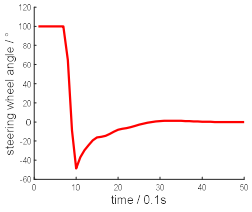
(e)Initial status



(f)Intelligent vehicle planning trajectory



(g)Heading angle of intelligent vehicle



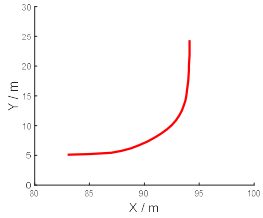
(h)Steering wheel angle control sequence

Figure 8. Lane retention experiment

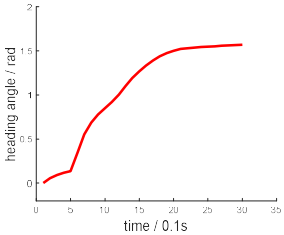
4.2.2. Curve Track Planning

During the simulation test, the intelligent vehicle sets out with different yaw angles, which are shown in Figure 9. Figure 9(b) and 9(f) have an abscissa of "X/m" and an ordinate of "Y/m". Figure 9(c) and 9(g) abscissa is "time/0.1s", ordinate is "heading angle/rad". Figure 9(d) and 9(h) abscissa is "time/0.1s" and the ordinate is "steering wheel angle/°". Although there is a shock in the curve, the intelligent vehicle still ultimately passes the curve successfully. Meanwhile, the curve of control sequence and the change rate of yaw angle are relatively smooth.

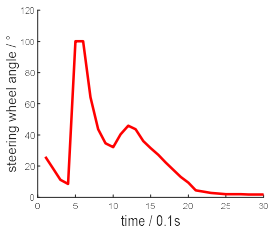
(a)Initial status



(b) Intelligent vehicle planning trajectory



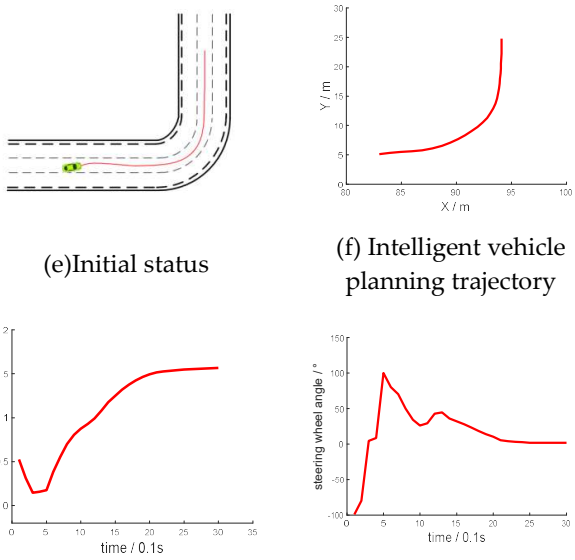
(c)Heading angle of intelligent vehicle



(d)Steering wheel angle control sequence

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905



(e)Initial status

(f) Intelligent vehicle planning trajectory

(g)Heading angle of intelligent vehicle

(h)Steering wheel angle control sequence

Figure 9. Curve driving experiment

4.3. Experimental Comparison and Analysis of the Three Trajectory Planning Methods

Nowadays, there are three trajectory planning methods comparison and experimental analysis in this section. Those trajectory planning methods are called, optimal trajectory planning method based on cubic polynomial, end-to-end trajectory planning method, and the model transfer trajectory planning method based on deep reinforcement learning. The intelligent vehicle drives following planning results in each simulation steps under the same tracking control error. Here, the experiments only compare and analyze the performance of planning. The traditional trajectory planning method adopts the angle control sequence based on preview window. However, the end-to-end trajectory planning method [28] outputs the angle control sequence directly. The period of the trajectory planning is 100ms. In other words, the intelligent vehicle is reprogrammed after keeping the same deflection angle for 100ms.

4.3.1. Arc Straight Track Scene

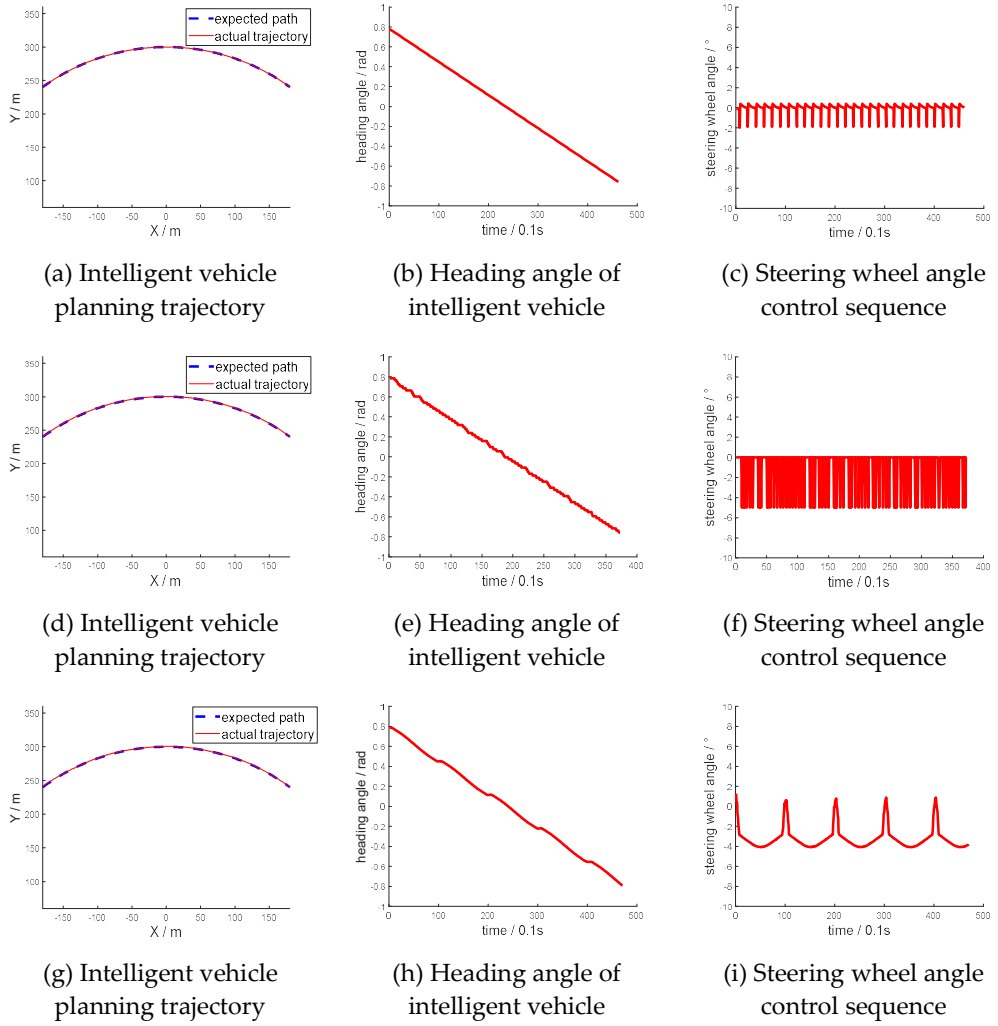
In Figure 10, the arc radius is 300 meters, so it is large enough to be seemed as a straight line in a small range. However, the curvature of is not equal to zero. The contrast experiment is shown in Figure 11. Figure 11(a)~(c) is the experimental results based on cubic polynomial dynamic optimal trajectory planning method, and Figure 11(d)~(f) is the experimental results of end-to-end trajectory planning method, Figure 11(g)~(i) is the experimental results of model transfer trajectory planning based on deep reinforcement learning. Figure 11(a), 11(d) and 11(g) have an abscissa of "X/m" and an ordinate of "Y/m". The blue dotted line in the Fig is "expected path" and the red solid line is "actual trajectory". Figure 11(b), 11(e), 11(h) abscissa is "time/0.1s", ordinate is "heading angle/rad". Figure 11(c), 11(f), 11(i) the abscissa is "time/0.1s" and the ordinate is "steering wheel angle/°".

The actual trajectory by the solid line in three methods are basically the same as the expected trajectory by the dotted line. But the rotation angle control sequence of the first two methods is oscillated continuously in the small range. The control sequence of our proposed method in this paper shows periodic and continuous variation, it is relative non-step oscillation. The experimental results verify that the model transfer trajectory planning method based on deep reinforcement learning performs well, and control action is continuity when it is driving nearly straightly.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

Figure 10. Arc straight track scene diagram



(a) Intelligent vehicle planning trajectory

(b) Heading angle of intelligent vehicle

(c) Steering wheel angle control sequence

(d) Intelligent vehicle planning trajectory

(e) Heading angle of intelligent vehicle

(f) Steering wheel angle control sequence

(g) Intelligent vehicle planning trajectory

(h) Heading angle of intelligent vehicle

(i) Steering wheel angle control sequence

Figure 11. Contrast experiment of Arc straight track scene

4.3.2. S-type Ramp Scene

Figure 12 is the schematic diagram of 'S' ramp. The intelligent vehicle is initially situated in the left lane of lower right corner. After driving at a yaw angle from the north, the vehicle turns left and enters 45°ramp. After driving a distance, it turns right in the middle lane and completes intelligent driving. The experiment results are shown in Figure 13. Figure 13(a)~(c) are the experimental results based on cubic polynomial dynamic optimal trajectory planning, Figure 13(d)~(f) are the experimental results of end-to-end trajectory planning based on depth network, and Figure 13(g)~(i) are the experimental results of model transfer trajectory planning based on deep reinforcement learning. The abscissa of Figure 13(a), 13(d) and 13(g) are "X/m", and an ordinate of Figure 13(a), 13(d) and 13(g) are "Y/m". The blue dotted line in Figure 13(a) is "expected path" and the red solid line is "actual trajectory". Figure 13(b), 13(e), 13(h) abscissa is "time/0.1s", ordinate is "heading angle/rad". The abscissa of Figure 13(c), 13(f), 13(i) are "time/0.1s", and their ordinate are "steering wheel angle/°".

According to the comparison results between actual trajectory and desired path based on those three planning methods, the results show that the proposed method maintains the intelligent vehicle

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 August 2018 doi:10.20944/preprints201808.0049.v1

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

a better turning performance, especially in straight and curve combined continuously roads. Finally, we gain that the lateral deviation of our proposed method is the least, and the performance outperforms the other two methods.

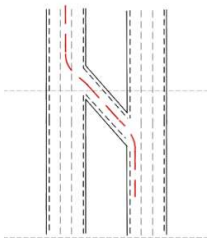
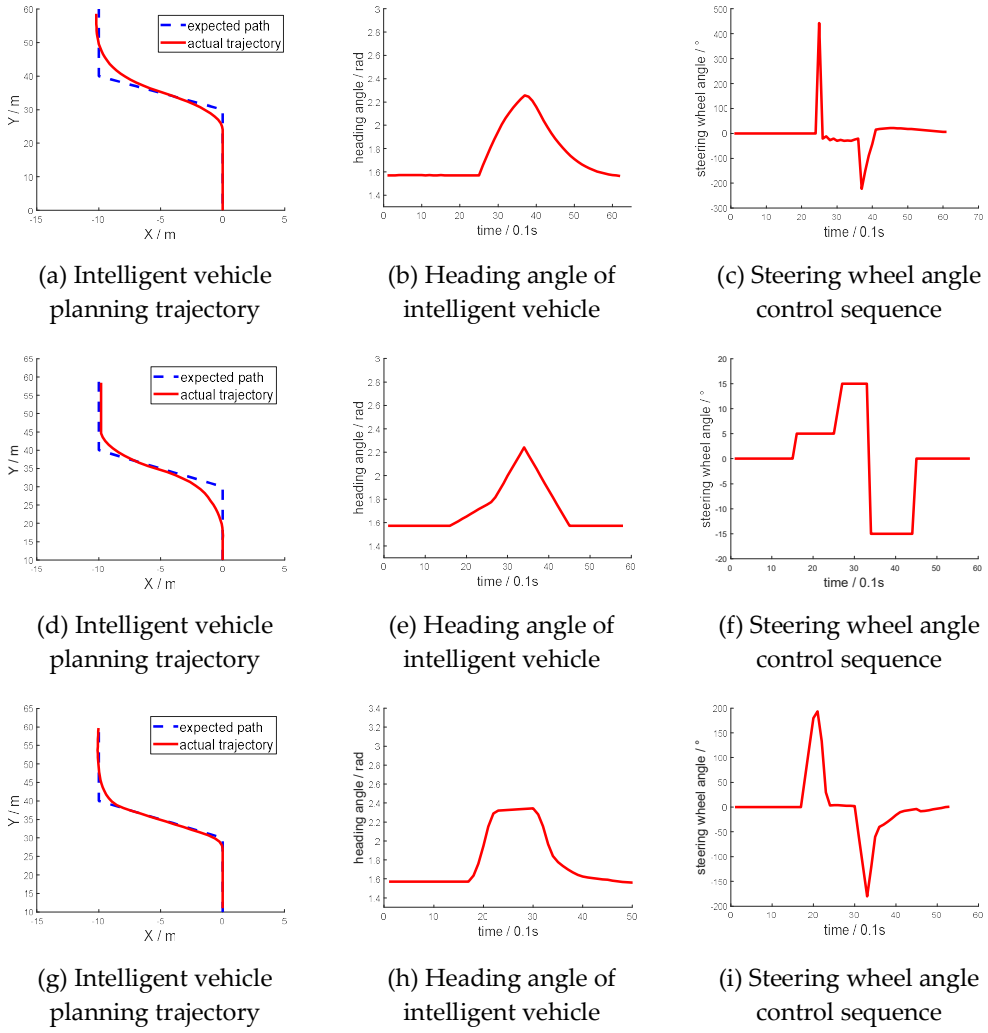


Figure 12. schematic diagram of S ramp



(a) Intelligent vehicle planning trajectory

(b) Heading angle of intelligent vehicle

(c) Steering wheel angle control sequence

(d) Intelligent vehicle planning trajectory

(e) Heading angle of intelligent vehicle

(f) Steering wheel angle control sequence

(g) Intelligent vehicle planning trajectory

(h) Heading angle of intelligent vehicle

(i) Steering wheel angle control sequence

Figure 13. Contrast experiment of S-type ramp scene

5. Real Vehicle Verification of Dynamic Trajectory Planning

The vehicle real-time position is collected by GPS/IMU inertial navigation system, The surroundings sensory data is acquired by camera, lidar, and millimeter-wave radar, which are mounted on the driverless vehicle. Figure 14 shows the actual vehicle dynamic trajectory planning process. Figure 14(a)~(i) are captured when the vehicle is turning in different scenes. Here Figure 14(a)~(f) are the outside view, while Figure 14(g)~(i) are the inside view of driverless vehicle. Figure

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905

14 shows that the driverless performance of intelligent vehicle is stable and efficient when it turns around 90°; Fig14.(j)~(o) are captured when the driverless vehicle is overtaking and lane changing. Figure 14(j)~(l) show the outside-view and Figure 14(m)~(o) show the internal view. And they indicate that the intelligent vehicle achieves self-overtaking and lane changing driving behaviors safety and stably. Figure 14(p)~(r) are real-time screenshot and interface of intelligent driving trajectory planning. Fig14(r) is the tracking result when the actual vehicle verifies the curve driving.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

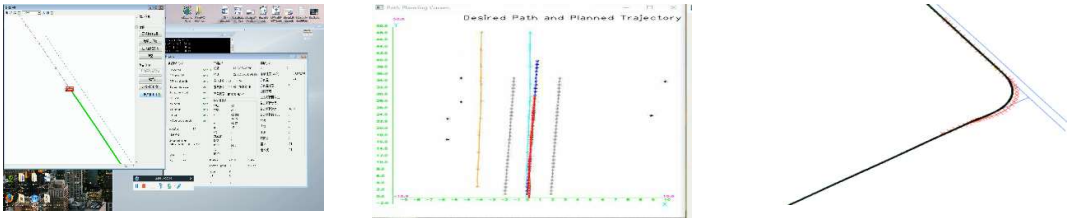
(l)

(m)

(n)

(o)

Peer-reviewed version available at Sensors 2018, 18, 2905; doi:10.3390/s18092905



(p) (q) (r)

Figure 14. Experimental results of real vehicle

6. Conclusions

It's different for traditional trajectory planning method to eliminate errors by vehicle models and road conditions. Furthermore, there are no vehicle dynamics constraints. A model transfer trajectory planning method based on deep reinforcement learning is proposed in this paper. At first, the complex real environment is abstracted by MTTP, then the abstracted model is transferred into a simple virtual environment through the transfer model. Secondly, the optimal intelligent driving maneuver after deep reinforcement learning training is applied to obtain the optimal control-trajectory sequence in the virtual environment. Thereby, end-to-end trajectory planning of intelligent vehicle in the real environment is realized. Meanwhile, an evaluation function is designed to estimate the planning validness of control-trajectory sequences, and to judge the risk of collision in real environment. Furthermore, an optimal control-trajectory sequence is decided and executed by the intelligent land vehicle. Finally, the comparison experiments analysis of multiple driving scenes and multiple trajectory planning methods verify the better optimization performance of MTTP, and that it brings a more continuous rotation angle control sequence and smaller lateral error for the intelligent land vehicle. However, the speed of vehicles is assumed as a constant and driving on a typical structured environment in this paper, the next stage will further consider variable speed driving and more complex environments.

Author Contributions: Conceptualization, Xuanya Shao and Yadong Wei.; Methodology, Xuanya Shao.; Software, Xuanya Shao and Yadong Wei.; Writing-Original Draft Preparation, Yadong Wei.; Writing-Review & Editing, Lingli Yu and Kaijun Zhou; Supervision, Lingli Yu and Kaijun Zhou.

Funding: This research was funded by Major Projects of Science and Technology in Hunan (Grant No.2017GK1010), National Key Research and Development Plan (Grant No. 2018YFB1201602), State Key Laboratory of Mechanical Transmissions of Chongqing University (SKLMT-KFKT-201602), State Key Laboratory of Robotics and System (HIT)(SKLRS-2017-KF-13), National Natural Science Foundation of China (Grant No.61403426) and Fundamental Research Funds for the Central Universities of Central South University (Grant No.2018zzts557).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wei K, Ren B. A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm[J]. Sensors, 2018, 18(2):571.
2. Coombes M, Fletcher T, Chen WH, Liu C. Optimal Polygon Decomposition for UAV Survey Coverage Path Planning in Wind[J]. Sensors, 2018, 18(7):2132.
3. Rastelli J P, Lattarulo R, Nashashibi F. Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles[C]. Proceedings of IEEE Intelligent Vehicles Symposium., 2014, 510-515.
4. Cong Y, Sawodny O, Chen H, et al. Motion planning for an autonomous vehicle driving on motorways by using flatness properties[C]. IEEE International Conference on Control Applications., 2010: 908-913.
5. Yu L, Long Z, Zhou K. Non-time trajectory tracking method based on Bezier curve for robot[J]. Chinese Journal of Scientific Instrument, 2016.

6. Sahingoz O K. Generation of Bezier Curve-Based Flyable Trajectories for Multi-UAV Systems with Parallel Genetic Algorithm[J]. Journal of Intelligent & Robotic Systems, 2014, 74(1-2):499-511.
7. Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587):484-489.
8. Santana E, Hotz G. Learning a Driving Simulator. arXiv: 1608.01230 [cs.LG], 2016, https://arxiv.org/abs/ 1608.01230v1.
9. Paxton, Chris; Raman, Vasumathi; Hager, Gregory D. et al. Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments[C]. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) , Sep. 24-28, 2017. Vancouver, Canada, 6059-6066.
10. Pfeiffer M, Schaeuble M, Nieto J, et al. From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots [C]. IEEE International Conference on Robotics and Automation (ICRA) 2017, 1527-1533.
11. Liu W, Li Z, Li L, et al. Parking Like a Human: A Direct Trajectory Planning Solution[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(12): 3388-3397.
12. Lin Y L, Li L, Dai X Y, et al. Master general parking skill via deep learning[C]. IEEE Intelligent Vehicles Symposium, 2017, 941-946.
13. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529.
14. Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2015, 8(6), A187.
15. Levine S, Levine S, Levine S, et al. Continuous deep Q-learning with model-based acceleration[C] International Conference on International Conference on Machine Learning.2016:2829-2838.
16. Schaul T, Horgan D, Gregor K, et al. Universal value function approximators[C] International Conference on Machine Learning. 2015: 1312-1320.
17. Metz L, Ibarz J, Jaitly N, et al. Discrete sequential prediction of continuous actions for deep RL[J]. arXiv preprint arXiv: 1705.05035, 2017.
18. Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay[J]. arXiv preprint arXiv:1511.05952, 2015.
19. Andrychowicz M, Wolski F, Ray A, et al. Hindsight experience replay[C]. Advances in Neural Information Processing Systems. 2017: 5048-5058.
20. Uhlenbeck G E, Ornstein L S. On the Theory of the Brownian Motion[J]. Physical Review, 1930, 17(2-3):323-342.
21. Plappert M, Houthooft R, Dhariwal P, et al. Parameter space noise for exploration. arXiv preprint arXiv:1706.01905, 2017.
22. Gu S, Holly E, Lillicrap T, et al. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates[C]. IEEE International Conference on Robotics and Automation (ICRA),2017: 3389-3396.
23. Genders W, Razavi S. Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142, 2016.
24. Isele D, Rahimi R, Cosgun A, et al. Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning. arXiv preprint arXiv:1705.01196, 2017.
25. Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation[C]. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017: 31-36.
26. Mirowski P, Pascanu R, Viola F, et al. Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673, 2016.
27. Kinga D, Adam J B. A method for stochastic optimization[C]. International Conference on Learning Representations (ICLR). 2015.

28. Wei Liu, Zhiheng Li, Li Li, Fei-Yue Wang, Parking like human: A direct trajectory planning solution[J]. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(12), pp. 3388-3397.