Lane-less Traffic Maneuvering using Deep Reinforcement Learning.

Jeevan Raajan, Dr.Ramkrishna Pasumarthy, IIT Madras

***Abstract*— In developing countries, vehicles seldom follow lane discipline and traffic scenario is termed chaotic. In such Lane-less traffic scenarios traditional lane model based approaches fail. A more robust approach is required for developing decision making and control models in such scenarios. Reinforcement learning has shown promise of handling decision making tasks in highly complex state spaces. The problem of lane-less traffic maneuvering is formulated under a MDP (Markov Decision Process) framework. A Reinforcement learning setup is considered and the agent is trained on the environment to improve its expected long term reward, wherein the reward is designed based on the required driving behaviours. We show that for a slow moving traffic in a broad highway, an observation window (Image) can be considered for the agent vehicle and a neural network can be trained to perform the task of vehicle control to maneuver in such lane-less traffic scenarios. Also the behaviour of the neural network based controller when placed in a Modelled Laned traffic is analysed.**

I. INTRODUCTION

Decision making and control in laned traffic has been well studied with the help of lane traffic models [11] [12]. However, Control of non-holonomic vehicle is still in the beginning stages in developing countries where lane-less traffic is predominant. Complex systems such as a non-holonomic vehicle are often difficult to model and are usually approximated by simpler model for building a controller. Also the environment considered can be complex for a lane-less traffic and the input to the controller can have a high dimensional state space which makes the design of a controller difficult. Often, in such systems where the objective is decision making and control, approximations in the model lead to adding constraints on the state space and the action space which can lead to a constrained implementation.

In our problem, we consider a non-holonomic vehicle driven in a laneless traffic scenario. We assume that the vehicles follow a Lane-less traffic model as defined in [1]. We assume that the environment consists of an infinite length highway where the drivers do not follow lane-discipline. Also the traffic is assumed to be a slow moving dense traffic. The agent vehicle is considered to be a simple Bicycle model. An illustration of the considered traffic scenario is provided in fig 1.

II. APPROACH: IMAGE BASED CONTROL

The image of the forward traffic up to 100 metres is considered to be available. This range is regarded as the observation window. At any point in time, only the positional information of the vehicles in the observation window is available and the intentions of these drivers are not known. The problem is thus modelled as a Partially Observable Markov Decision Process (POMDP). The Intentions of the drivers can however be observed if we sample multiple image readings over time and extract the rate and direction of movement of the surrounding vehicles. This concept is utilized in building a convolution neural network which can analyse temporally placed sequence of images. The Convolution Neural Network (CNN) is used to extract state features from the sequence of images and this is followed by a multi-layer perceptron layer which builds the Q-function from the features extracted from the CNN layer. This Q-Network is then trained to maximize the expected long term reward.

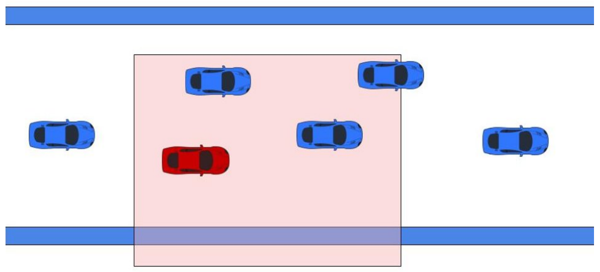
Fig. 1. Lane-less Traffic scenario with observational window

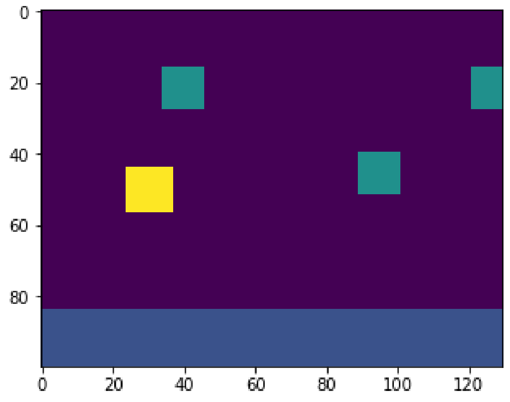

Fig. 2. Pre-processed image after re-sizing and normalization. The yellow block represents the agent vehicle. The blue blocks represent the other vehicles and the road edges.

III. MICROSCOPIC MODEL ENVIRONMENT FOR SIMULATING THE LANE-LESS TRAFFIC:

A Microscopic Model for Lane-Less Traffic is used to simulate the traffic network.

A. Assumptions:

1) The X-dynamics and Y-dynamics are uncoupled and the drivers can accelerate independently in the two orthogonal directions.
2) The preferred longitudinal velocity of the vehicles in the homogeneous set are modelled by a fictitious vehicles a_0^y leading the convoy.
3) The 2 road edges are considered in the X-dynamics 2 vehicles of fixed positions and zero velocity and are represented as a_0^x and a_{n+1}^x having $x_0 = 0$ and $x_{n+1} = d$ respectively.
4) The Y-Dynamics has unidirectional influence ,whereas the X-Dynamics has bidirectional influence.
5) The angle of viewing (aov) in lateral direction is modelled as two sideways symmetric cones, whereas the angle of viewing in longitudinal (Y-axis) direction is modelled as a single cone placed in front of the vehicle.

B. Y-Dynamics(Longitudinal):

Consider a set of k vehicles belonging to a homogeneous set $\mathcal{M}$. Let $y = [y_0\ y_1\ ....y_k]^T \in \mathbb{R}^{k+1}$ and $v_y = [v_{y0}\ v_{y1}\ ....v_{yk}]^T \in \mathbb{R}^{k+1}$ be the positions and velocities of these vehicles. Each vehicle is modelled as follows.

$$\dot{y_i} = v_{yi} \tag{1}$$

$$\dot{v_{yi}} = \sum_{j\in\mathcal{N}_i} -\left(b_y w_{ij}(v_{yj} - v_{yi}) - k_y\left(w_{ij}(y_j - y_i) + \frac{1}{|\mathcal{N}_i|}(g_y - \bar{k}v_i)\right)\right) \tag{2}$$

The terms are explained as given below:

1) y_i: $Y-$position of the i^{th} vehicle
2) v_{yi}: $Y-$velocity of the i^{th} vehicle
3) b_y: Denotes the influence of the relative velocities between the vehicles on acceleration.
4) k_y: Denotes the influence of the relative spacings between the vehicles on acceleration.
5) $|\mathcal{N}_i|$:Number of vehicles present in the immediate next layer of the visibility cone of angle defined by aov. In our case aov is chosen to be 90^o
6) w_{ij}: are the edge weights and are defined in our environment as the influence of the node a_j with respect to the node a_i.We have $\sum_{j=1}^{n} w_{ij} = 1$ ie. The cumulative wights of incoming edges.we consider the weights with respect to each car equal and hence we have $w_{ij} = \frac{1}{|\mathcal{N}_i|}$

C. X-Dynamics(Lateral):

The road edges are considered as vehicles a_0^x and a_{n+1}^x.

$$\dot{x_i} = v_{xi} \tag{3}$$

$$\dot{v_{xi}} = \sum_{j\in\mathcal{N}_i} -\left(b_x w_{ij}(v_{xj} - v_{xi}) - k_x w_{ij}(x_j - x_i)\right) \tag{4}$$

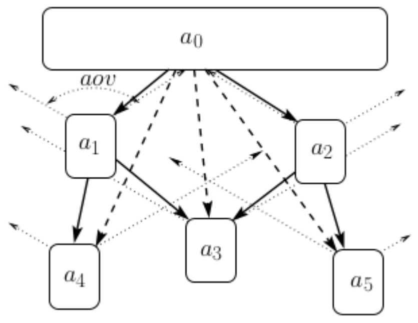

Fig. 3. Y-influence graph

The constants are similar to as explained for Y-Dynamics. Intuitively, each vehicle tries to go to the midpoint of the weighted average of X-positions of its left and right neighbours and to maintain the safe distance from vehicles both on the left and the right.

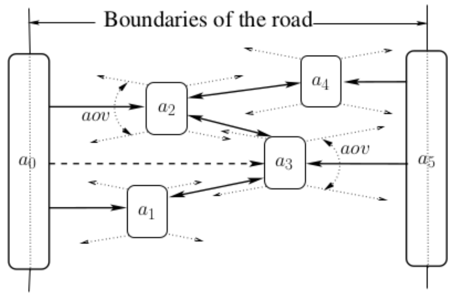

Fig. 4. X-influence graph

IV. RL Agent:

A simplified bicycle model as shown in fig5 is used [10] and the dynamics are described as follows:

$$\dot{x} = vcos\theta$$
$$\dot{y} = vsin\theta$$
$$\dot{\theta} = \frac{vtan\psi}{l}$$

V. MDP formulation:

MDP definition:An MDP satisfies the Markov property, which means that the probability distribution of the future states depends only on the current state and action, and not on the history of previous states.

Since the intention of other road users cannot be observed, the speed and direction change decision making problem can be modeled as a partially observable Markov decision process (POMDP). To address the partial observability, the POMDP can be approximated by an MDP with a k-Markov approximation, where the state consists of the last k observations, st =$(o_{t-k+1}, o_{t-k+2}, ..., o_t)$ We have considered k=3.

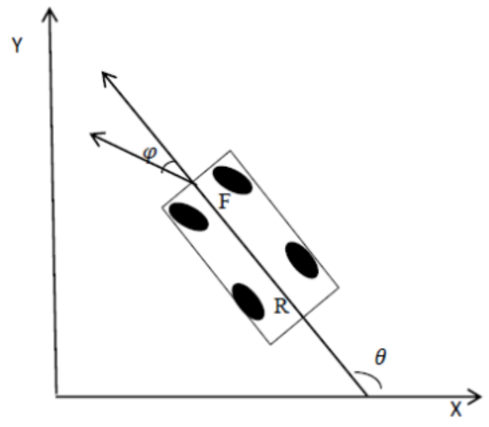

Fig. 5. Simplified Bicycle Model

A. States

The image of the observational window is pre-processed by normalizing it and re sizing it to 100×130 pixels. This image is then stacked along with the previous 2 timestep images. The 3 images then form a 3 channel image of size $100 \times 130 \times 3$.

B. Actions

The action space is the set, $a_\theta \times a_f \in \mathbb{R}^2$ ie.$\big[-1,1\big] \times \big[-2,2\big] \in \mathbb{R}^2$. where a_f is the forward acceleration and a_θ is the radial acceleration. The decision making interval is set as $\Delta t = 1$ second

Value	Interpretation
$\big[-1,0\big]$	Steer left
$\big[1,0\big]$	Steer right
$\big[0,0\big]$	Do nothing
$\big[0,1\big]$	Accelerate
$\big[0,-2\big]$	Decelerate

TABLE I
ACTION SPACE

C. Reward

The reward function is used to learn the required driving behaviour.

1) To drive the vehicle forward, At every time step, a positive reward is given, proportional to the y-directional distance driven during that interval, $c * \Delta d$, and normalized as, $c * \Delta d/d_{max} = c * \Delta t v_y^{ego}/v_{max}^{ego}$, where v_{max}^{ego} is the maximum possible speed of the ego vehicle.

$$r_{vel} = k_{vel} \; v_y^{ego}/v_{y-max}^{ego}$$

2) To reduce unnecessary steering, a penalty proportional to the angular deviation (θ) of the agent vehicle from the parallel of the road is given at every step.

$$r_{dev} = -k_{steer}|\theta|$$

3) To provide a safe distance while driving, a penalty inversely proportional to distance of the surrounding cars within 20 metres of the agent car is given.

$$r_{safe} = \sum_{i=0}^{n} k_{safe}(20 - d_i), d_i \leq 20$$

where n is the number of cars within 20 metres distance and epsilon is small quantity set to avoid division by zero.

4) A large penalty is given for collision with the road edges or other vehicles.

$$r_{collide} = -5$$

At any step, the Total reward is the sum of all the above rewards. The velocity part of the reward function implicitly encourages the ego vehicle to overtake slower vehicles. The penalty for maintaining a safe distance ensures the agent drives as with some safety gap and the penalty for deviating from the parallel of the road ensures stable driving. However, if a collision occurred, or the ego vehicle drives out of the road, a penalizing reward of -5 is given and the episode is terminated.

$$R = \begin{cases} -10 & \text{if collision occurs} \\ r_{vel}^{y} + r_{dev} + r_{safedis} + r_{collide} & \text{otherwise} \end{cases}$$

VI. Deep Q-Learning:

A. Reinforcement learning:

Reinforcement learning is a branch of machine learning where in the objective is to find an optimal policy, π^*. The optimal policy maximizes the cumulative reward which is received by exploring the environment through different actions. The Q-value, $Q(s, a)$ is defined as the function which returns the expected future reward from a state s, for an action a, following policy π. In order to effectively deal with high dimensional state spaces, we use the Neural Network as a function approximator to estimate the Q-values for a state-action pair. The neural network is a non-linear function, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where n and m are the dimension of the state space and action space respectively. The advantage of using a neural network is the ability to generalize to a new environment through a known environment. The environment state is fed as an input to a neural network which then outputs the action-value pairs. Initially the environment is explored by taking random actions and the experience gathered is used to train the neural network. The amount of exploration is slowly decayed and the environment is then exploited for high rewards by using a greedy strategy to select the optimal action.

B. Deep Q-Network

In Q-learning, the optimal Q-function is learned which maximizes the expected future Return for a state-action pair:

$$Q^*(s, a) = \max_\pi \mathbb{E}_{s,a\sim\pi}[R_t|\pi] \qquad (5)$$

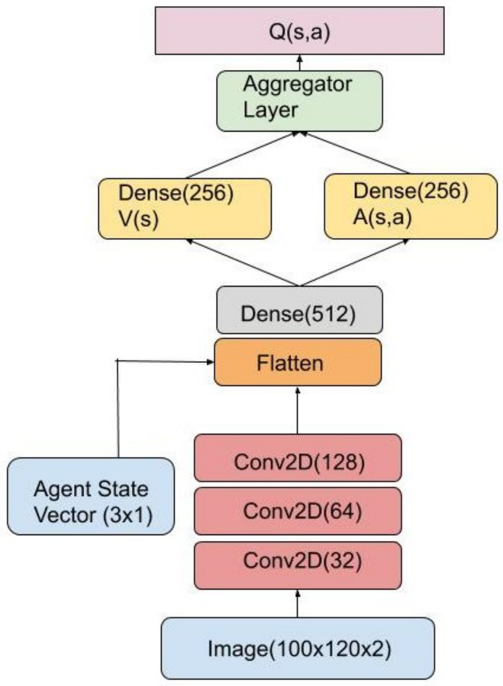

Fig. 6. Q-Network Architecture

The Bellman equation is used to learn the Q-function. The optimal Q-function satisfying Bellman equation is defined as

$$Q^*(s,a) = \mathbb{E}_{s,a\sim\pi,r}[r + \gamma \text{max}_{a'} Q^*(s',a')|s,a] \quad (6)$$

If y_i is the target Q-value and $Q(s,a;\theta_i)$ is the Q-value predicted by the neural network, the loss function to be minimized is defined as:

$$L_i = (\theta_i) = \mathbb{E}[(y_i - Q(s,a,;\theta_i))^2] \quad (7)$$

where y_i is given as

$$y_i = \mathbb{E}[r + \gamma \text{max}'_a Q(s',a';\theta_{i-1}|s,a)] \quad (8)$$

The neural network is then trained by adjusting its parameters θ at each iteration i to minimize the loss function. Backpropagation with Adam optimizer is used to perform this at every iteration i.

C. Neural Network Design

The table below describes the CNN architecture used for designing the Q-Network.

However the states of the agent vehicle, namely v, θ, ψ cannot be passed in the 1st layer along with the image states. We accommodate these states by concatenating them in the hidden dense layer following the first 3 CNN layers states.

This architecture allows us to combine both image based data as well individual state data to the network.

The aim of the neural network is to extract positional and velocity information of the surrounding vehicles in the observation window and output a Q-Function which can be used to predict the control action to give a desired driving behaviour as designed by the reward function.

Layer	Topology	Activation
1	Convolution ,32 8x8 kernels with stride 4	elu
2	Convolution ,64 4x4 kernels with stride 4	elu
3	Convolution ,128 4x4 kernels with stride 2	elu
4	Dense, concat(512 neurons,3 Inputs)	elu
5	Dense 256, Dense 256	elu
6	1 neuron- V-Function,5 neurons- Adv function	Linear
7	Dense 5 neurons- Q(s,a)	Linear

TABLE II
NEURAL NETWORK ARCHITECTURE

VII. SIMULATION SETUP:

1) **Infinte length Highway:**The state space does not provide information on where the agent is at a given time step. eg: the beginning or close to the end. This is done because the objective was to drive on an infinite length highway. In addition to the states, this is done by discarding the end state where the agent reaches the end of the longitudinal section of the simulated highway, since the future expected discounted return, R_{end} ia 0. This way, all the experiences stored in the replay buffer trick the agent into believing that the episode continued forever.
2) **Fictitious leader vehicle**:The longitudinal velocity of the leader vehicle is simulated with a fixed component of Y-velocity added with Gaussian random noise to make it more practical.ie $v_{y0} = c + \mathcal{N}(0,\sigma)$, where the constant c and sigma are varied with each episode to simulate traffic scenarios of different traffic speed. This is done so that the agent is capable of driving in various traffic scenarios.
3) **Vehicles belonging to the homogeneous set**: Each vehicle follows the modelled equation
4) **Influence graph**$-\vec{\mathcal{G}}^{y}_{cone}$: contains the vehicles in the next assigned level in the visibility cone. Each level is assigned a longitudinal range of 30 metres. This means,all vehicles in the 30 metre longitudinal range are assigned in the same level. This is done as follows: If there exist a vehicle which is leading a particular level and there is a vehicle behind which is more than 30 metres behind, it is then assigned a new level. If the following vehicle is within 30 metres of the leading vehicle, then it assigned the same level of the leading vehicle. When multiple vehicles are present behind a level leader. The level leader is taken as the reference for level calculation for the vehicles behind. Level range=30 metres
5) A total of 10 vehicles are used in the simulation. 1 of them is the fictitious leader vehicle. 8 are the follower vehicles and the last one is the agent vehicle.
6) The dimensions used are:
   - Road width=400
   - Road length=2000
   - Vehicle =square of sides 50
7) The initial conditions(positions and velocities) of the vehicles are changed with each episode to make the

RL agent adaptive to different conditions and are initialized as follows:

- ie. The leader is simulated to be the front most vehicle during each episode. But, since this is fictitious vehicle it does not affect the RL agent.
$$x_0 = 200, y_0 \sim \mathcal{U}(500, 1500)$$
- Let g_p be the minimum inter-vehicle distance when initializing the positions at the start of an episode, $a_i = [a_i, y_i]$ be the position of the a_i^{th} vehicle and w be the road width. x_i is the lateral position and y_i is the longitudinal position.The vehicles are initialized with the distribution for each episode as:
$$\left\{ x_i \sim \mathcal{U}(60, w-60),\ y_i \sim \mathcal{U}(50, y_0 - 50)\ \forall i = 0..k \middle| ||a_i - a_j|| > g_p \right\}$$
ie. every vehicle is initialized randomly with some minimum inter-vehicle gap to avoid collision and have sufficient space to stabilize their positions.
- $\left\{ v_{xi} \sim \mathcal{U}(0, w-3),\ v_{yi} \sim \mathcal{U}(0, 10) \middle| \forall i = 0...k \right\}$

8) Traffic parameter settings in Y and X-Dynamics equations:
   - $b_y = -0.02$
   - $k_y = 0.04$
   - $g_y = -100$
   - $\tilde{k} = 50$
   - $b_x = 0.04$
   - $k_x = 0.02$

VIII. To be incorporated:

1) **Collision considerations and overall stability of the system:**Currently we have only considered that ,the episode ends when the ego vehicle crashes. However ,the ego vehicle can influence other vehicle due to its movement and can cause collision between other vehicles. Hence we need to consider the episode to be a failure if any of the vehicles in the traffic system crashes.
2) **Traffic model:**
   a) The traffic model can be modified to include coupling between longitudinal and lateral motion.The consideration here is that the longitudinal acceleration of the vehicle is reduced when there is a lateral movement.This is done by modifying the equation as follows:

$$\dot{y_i} = v_{yi} \tag{9}$$

$$\dot{v_{yi}} = \sum_{j \in \mathcal{N}_i} -\left(b_y w_{ij}(v_{yj} - v_{yi}) - k_y \left(w_{ij}(y_j - y_i) + \frac{1}{|\mathcal{N}_i|}(g_y - \bar{k} v_i) \right) \right) - m(sign(v_{xi}) v_{xi}) \tag{10}$$

for some m¿0

3) **Add non-homogeneous vehicle** : This model allows the agent vehicle to optimize itself to pass through the edges of the road. To overcome this, we add a IDM(Intelligent driver model) based model to the existing traffic model where vehicles control is only longitudinal and are placed randomly including the edges of the road. This forces the agent to build a policy which not only considers a homogeneous set of longitudinally and laterally responsive vehicles, but also considers a heterogeneous set of vehicles which are responsive only longitudinally.

IX. Simulation Results

References

[1] Everett, Michael, Yu Fan Chen, and Jonathan P. How. ”Motion planning among dynamic, decision-making agents with deep reinforcement learning.” 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[2] Duguleana, Mihai, and Gheorghe Mogan. ”Neural networks based reinforcement learning for mobile robots obstacle avoidance.” Expert Systems with Applications 62 (2016): 104-115.

[3] Pete Shinners and team(2011). PyGame - Python Game Development

[4] Mnih, Volodymyr, et al. ”Human-level control through deep reinforcement learning.” Nature 518.7540 (2015): 529.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, ”Deep Reinforcement Learning: A Brief Survey,” in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017.

[6] Van Hasselt, Hado, Arthur Guez, and David Silver. ”Deep reinforcement learning with double q-learning.” Thirtieth AAAI Conference on Artificial Intelligence. 2016.

[7] Lillicrap, Timothy P., et al. ”Continuous control with deep reinforcement learning.” arXiv preprint arXiv:1509.02971 (2015).

[8] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16), Maria Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. JMLR.org 1995-2003.

[9] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In International Conference on Learning Representations, 2016

[10] Samani, N. N., Ghaisari, J., Danesh, M. (2012, October). Autonomous parallel parking of a vehicle in a limited space using a RBF network and a feedback linearization controller. In 2012 2nd International eConference on Computer and Knowledge Engineering (ICCKE) (pp. 117-122). IEEE.

[11] Kesting, Arne, Martin Treiber, and Dirk Helbing. ”General lane-changing model MOBIL for car-following models.” Transportation Research Record 1999.1 (2007): 86-94.

[12] Treiber, Martin, Ansgar Hennecke, and Dirk Helbing. ”Congested traffic states in empirical observations and microscopic simulations.” Physical review E 62.2 (2000): 1805.