Notes for Deep Learning

TIANFU HE, School of Computer Science, Harbin Institution of Technology.

Additional Key Words and Phrases: Deep Learning, Reinforcement Learning

1 STORY OF CARS

Since the time my father bought his car, Geely EC7, I drive it sometimes. Gradually I get insterested in cars, especially the development of automotive industry in China, because it's a crutial step for Made-In-China becomes strong. And in this section, I write down the stories of me with cars.

1.1 A Game for Parking Exercise

How can the experienced drives park their cars so fast with out going back-forth? How do they get their cars into the very limited parking space among the crowd? The answer is, the intuition! If we get on well with our cars, we know how steering angle can change the direction, and finally we can feel the car's body as ours, but to achive that, we need training, a visual game in vertical view to help us get such intuition!

To this end I make a tool for parking exercise [1], with source code [2]. **The real-scenario steering behavior of the car is implemented**, i.e., by setting the angel of front-inner wheel, the simulated car will follow the trajectory of the real-world **2WD** cars.

I here introduces the mechanism of 2WD cars' movement, and provide the postion update function of cars given status of the car.

1.1.1 Mechanism of 2WD Cars' Movement. For a 2WD car like Figure 1a, when steering, only the front 2 wheels P_0, P_1 are under control, while the back wheels P_2, P_3 are always parallel to the car body. To keep the consistancy of four wheels when steering, the car should keep all wheels going circles with the same circle center C. So when the wheel at P_0 gets an angle θ, the position of circle center C is:

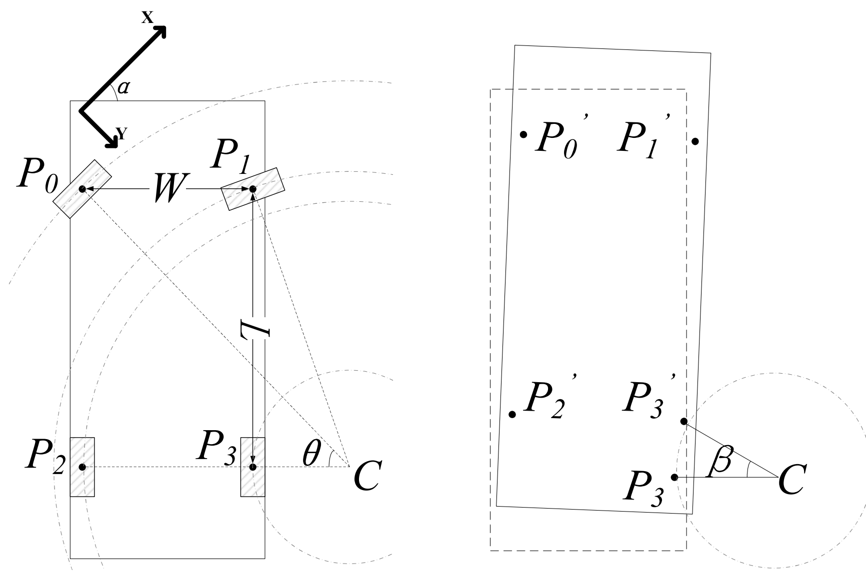
$$C = P_2 + \frac{L}{W \cdot \tan\theta}\overrightarrow{P_2P_3}, \tag{1}$$

where L and W is the wheel base and the inter-width between P_2 and P_3.

This work was done when Tianfu was on his way learning Machine Learning Techniques.

Author's address: Tianfu He, School of Computer Science, Harbin Institution of Technology. Tianfu.D.He@outlook.com.

XXXX-XXXX/2018/8-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Notations

(b) Moving.

Fig. 1. A 2WD Car.

1.1.2 The Update Function of Movement. How the postions of four wheels update when the car moves forward a bit? The following gives the update function when P_3 moves Δ **units** on its circle.

$$
\begin{aligned}
|\overrightarrow{CP_3}| &= L/\tan\theta - W \\
\beta &= \frac{\Delta}{|\overrightarrow{CP_3}| - W} \\
P_3' &= C + \cos\beta \cdot \overrightarrow{CP_3} + \frac{|\overrightarrow{CP_3}| \cdot \sin\beta}{L} \cdot \overrightarrow{P_2P_0} \\
P_2' &= C + \cos\beta \cdot \overrightarrow{CP_2} + \frac{\sin\beta}{\tan\theta} \cdot \overrightarrow{P_2P_0} \\
\alpha' &= \begin{cases} \arccos \overrightarrow{P_2'P_3'}, & \overrightarrow{P_2'P_3'} \cdot \widehat{y} > 0 \\ 2\pi - \arccos \overrightarrow{P_2'P_3'}, & \text{else} \end{cases} \\
P_0' &= P_2' + L \cdot \widehat{\alpha' - \frac{\pi}{2}} \\
P_1' &= P_0' + \overrightarrow{P_2'P_3'}
\end{aligned}
\tag{2}
$$

where $\widehat{\square}$ is the unit vector. C is calculated by Equation 1. (P_0', P_1', P_2', P_3') are the updated positions. β is the angle $\angle P_3CP_3'$. α is the rotation of $\overrightarrow{P_0P_1}$ to x-axis, and α' is the rotation of $\overrightarrow{P_0'P_1'}$ to x-axis.

1.1.3 The Implementation of 2WD Car in Program. With the equation given by Equation , we can easily implement the parking exercise game. The game keeps three types of variables: 1) Steering angle θ, 2) Car's direction α; and 3) Car's postion P_0, P_1, P_2, P_3. During the game, the player controls the car by telling it either *left/right*, which changes α, or *forward/back*, which triggers a Δ value for position updating.

To discretize the procedure of driving, we have the following assumption, without loss of gaming experience:

ASSUMPTION 1.1. *Even when the player keeps steering, for each iteration of position updating, the steering angle variable θ is a constant value, given that the movement Δ is small enough.*

Based on Assumption 1.1.3, we can set two event handler, `SteeringHanlder` and `MovingHanlder`, to support user's actions, and update the game asynchronously.

Algorithm 1 Moving Hanlder

Input: Moving distance of P_3, Δ.
Output: NULL.

$\theta_0 \leftarrow \theta.copy()$
Compute $(\alpha', P_0', P_1', P_2', P_3')$ by Equation 1.1.4 with params (θ_0, Δ)
Update α and the position by $(\alpha', P_0', P_1', P_2', P_3')$

We skip the description of `SteeringHanlder` since it simply update θ variable and is easy to implement. Algorithm 1 gives the detailed procedure of moving. First, to avoid θ changed by an asynchronous procedure of θ, the current θ value is copied. Then the updated direction α' and the positions of four wheels are calculated by Equation 1.1.4, by which we update the car's status in the game.

1.1.4 More Realistic: Acceleration, car body, etc. To make the game fun, we can design car body, add acceleration property to it, or something else. I didn't implement all of them, but it's still very easy.

Acceleration. With Equation , we can control the frequency of the updating, or set small or large Δ value to control the speed.

Car body. Since we have the update function of four wheels already, and the car body is relatively static to the four wheels, you can update the position of the car body by simple calculation.

REFERENCES

[1] [n. d.]. Parking Exercise Game. http://tianfudhe.coding.me/parking-exercise.html. ([n. d.]). Accessed: 2018-08-28.
[2] [n. d.]. Tianfu's Parking Exercise Source Code. https://coding.net/u/tianfudhe/p/tianfudhe.coding.me/git/raw/master/parking-exercise.html. ([n. d.]). Accessed: 2018-08-28.