**DRL在ADS中的分析与Torcs中的应用举例**

深度学习在图像识别、语义分析等很多方面已经取得了非常突出的成果，也促使了更多领域的发展，强化学习通过使用神经网络的逼近后发展成了一个新名词：DRL。在前两篇介绍RL和DRL基础上，本文重点介绍wu深度强化学习在自动驾驶模拟系统中的应用场景。



# 1、汽车平台及原理分析

控制平台是无人车的核心部件，控制着车辆的各种控制系统，包括汽车防抱死制动系统（ABS）、汽车驱动防滑转系统（ASR）、汽车电子稳定程序（ESP）、电子感应制动控制系统（SBC）、电子制动力分配（EBD）、辅助制动系统（BAS）、安全气囊（SRS）和汽车雷达防碰撞系统、电控自动变速器（EAT）、无级变速器（CVT）、巡航控制系统（CCS）、电子控制悬架（ECS）、电控动力转向系统（EPS）等等。控制平台主要包括了电子控制单元ECU与通信总线两大部分。（注：更多汽车相关的一些术语，建议大家看看汽车之家：

https://car.autohome.com.cn/shuyu/detail\_8\_10\_244.html）

（1）ECU(Electronic Control Unit): 电子控制单元，又称“行车电脑”、“车载电脑”等。从用途上讲则是汽车专用微机控制器。它和普通的电脑一样,由微处理器（CPU）、存储器（ROM、、RAM）、输入/输出接口（I/O）、模数转换器（A/D）以及整形、驱动等大规模集成电路组成。**用一句简单的话来形容就是“ECU就是汽车的大脑”，**其主要用于实现控制算法。

（2）通信总线（CAN: Control Area Network）：其实是一种控制局域网，它由德国BOSCH开发的用于[计算机控制系统](https://baike.baidu.com/item/%E6%B1%BD%E8%BD%A6%E8%AE%A1%E7%AE%97%E6%9C%BA%E6%8E%A7%E5%88%B6%E7%B3%BB%E7%BB%9F)和嵌入式工业控制局域网的标准总线，其在汽车领域主要实现ECU以及机械部件间等一系列通信功能 。

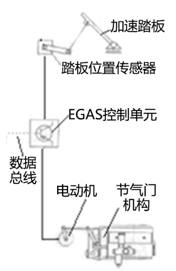
好了，上面说了汽车控制平台的两大核心ECU和通信，但是我们通过程序是怎样控制汽车了，难道要通过机器人去踩油门？握方向盘？（这和人开有什么区别呀），显然该方案理论可以，但并不是ADS追求的方案， **通常，**工业接会采用一种叫**线控驱动**（Drive-by-wire，通常俗称：线控）的电子技术，利用该技术直接通过应用程序去操纵汽车的油门系统、转向系统、刹车系统。

1. **线控油门：**

线控电子油门的主要功能是把驾驶员踩下油门踏板的角度转换成与其成正比的电压信号，同时把油门踏板的各种特殊位置制成接触开关，把怠速、高负荷、加减速等发动机工况变成电脉冲信号输送给电控发动机的控制器 ECU，以达到供油、喷油与变速等的优化自动控制，如图，因此我们只需要通过程序来操纵具体的电压信号，其可归结为：

ΔVt=αΔV

注：Vt相当于发动机转速，并不是直接速度，中间需要很多复杂转换

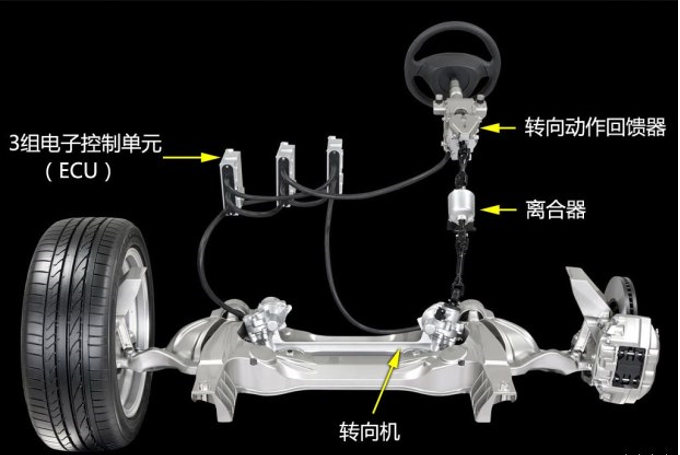


1. **线控转向系统**

下图是英菲尼迪的结构图展示，线控转向取消了方向盘与车轮之间的机械连接，用传感器获得方向盘的转角数据，然后ECU（电子计算单元）将其折算为具体的驱动力数据，用电机推动转向机转动车轮，表示为：

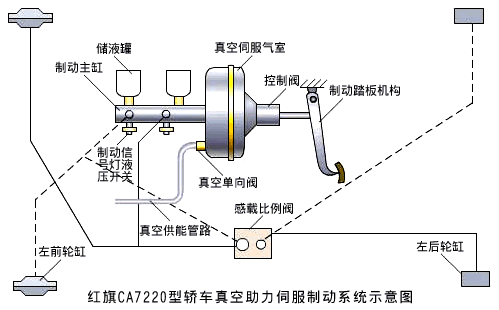
ΔFt=αΔθt

注：驱动力与转角的关系，同样中间通过复杂的变化



（3）线控制动系统

传统的制动系统中有很多的相关术语，包括ABS，ESP等，同样的道理，制动刹车板的角度与制动之间有对应函数关系，中间须通过助力泵等很多传感器设备等的协作完成。本文只对顶层进行概述。



更多关于怎么推导之间的变换关系的见文献[3]

**至此，本文对汽车最主要的三个控制元件进行了简单的描述。说了这么多，其实就是普及一下汽车的常识，下面将开始自动驾驶中使用的设备的简介。**

# 2、传感器组成及分析

**人开车过程场景： 注意力集中，两眼直视前方，手握方向盘，脚放在制动设备……**

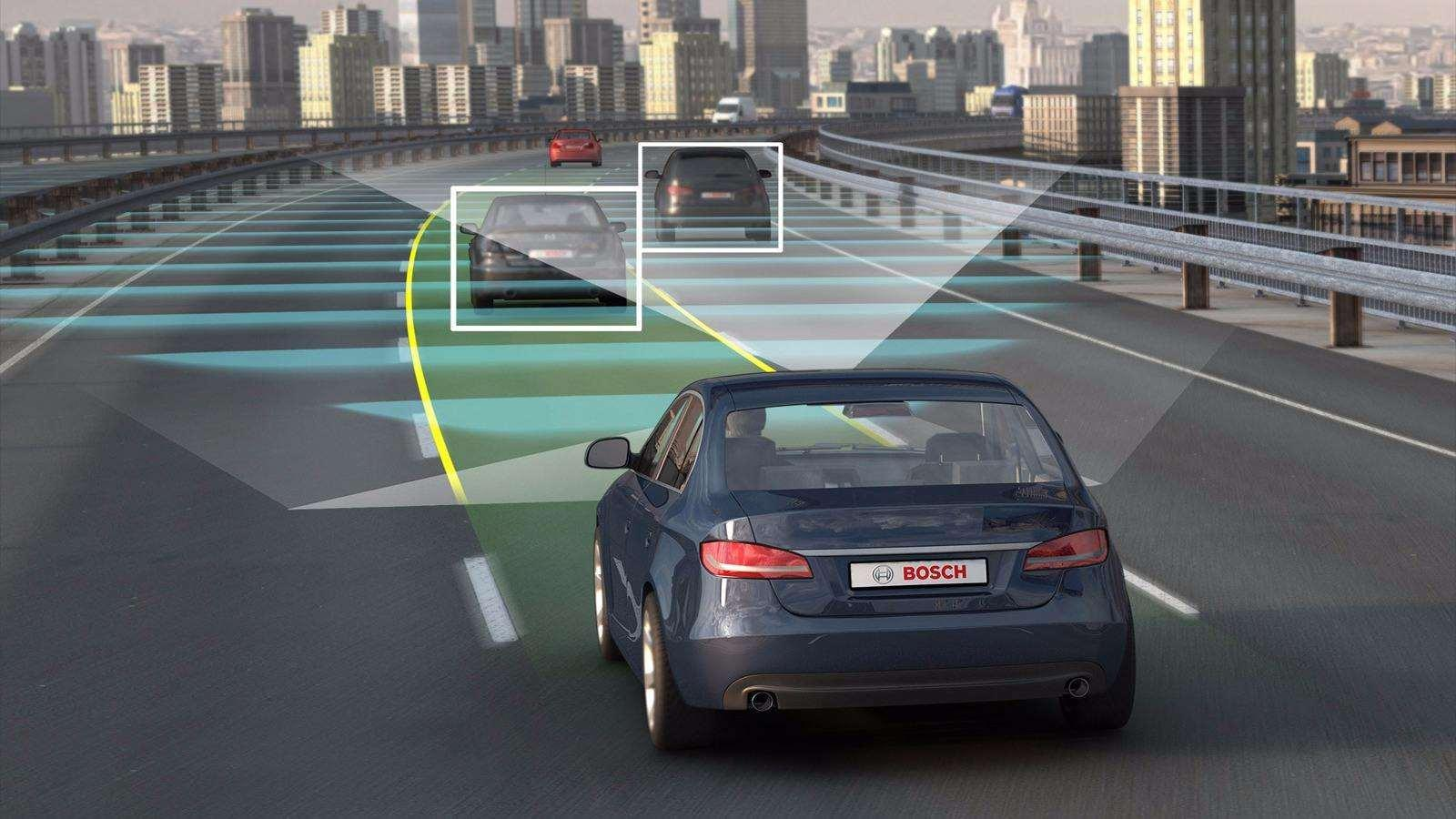
其实ADS和人开车原理是基本类似的，那么首先就需要一个眼睛：主要用来观察前方场景，两只眼睛可以感知距离(另外：ADS需要激光雷达LIDAR，GPS模块)，使用这两个主要是实现高精度定位和距离、物体检测。

好了，不扯了，是时候看看自动驾驶汽车的真面目了！【别人家的车，偷笑……】



1. **摄像头**

主要用于检测汽车前方的环境场景，行车车道线等，可以处理一些复杂的情况，语义分割，目标检测等



1. **GPS模块**

ADS目前还是非常依赖于高精度地图（三维立体） 左图:,采集方法和常用的地图采集原理（右图）基本类似，但是是经过三维建模后的空间三维地图。



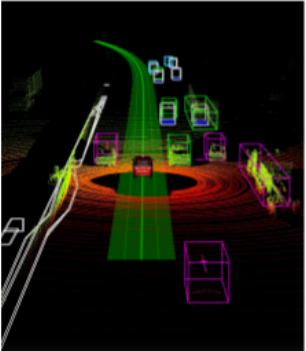
1. **激光雷达**



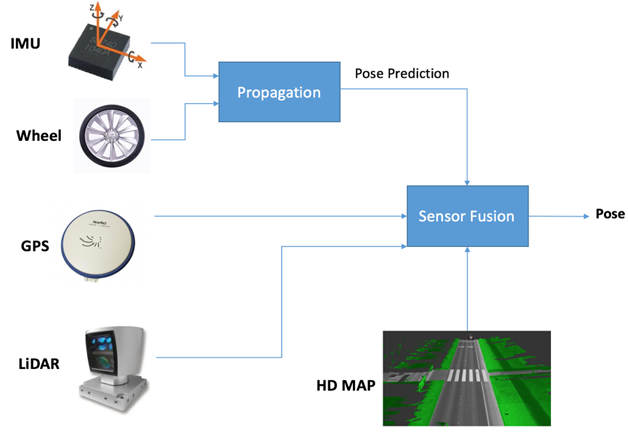
激光雷达的主要作用就是根据三维地图，以及对实际物体距离的毫米级距离检测

毕竟路边的基础设施不是你随便就能撞的，至于激光雷达的32线，64线自己google，具体区别在于精度等方面

那么激光雷达下的显示场景是什么样的呢？其实只是显示三维的框框实体，它并不知道是什么，只是能显示距离（建议：可以搞搞人体建模，对人进行识别，提高安全等级，毕竟从优先级上来说，宁可撞电线杆，也不撞倒行人）



接下来需要对汽车和传感器之间进行一个连接了，用一张图说话：



现在需要一个控制器，常见的是（工控机（Industrial PC，IPC）,但是略贵，最好破解汽车系统，并不简单！（当然可以采用apollo）



**So so 路漫漫其修远兮………**

# 3、汽车与传感器结合分析

ADS系统是一个非常复杂的工程系统，主要包括三层：感知，规划和控制。

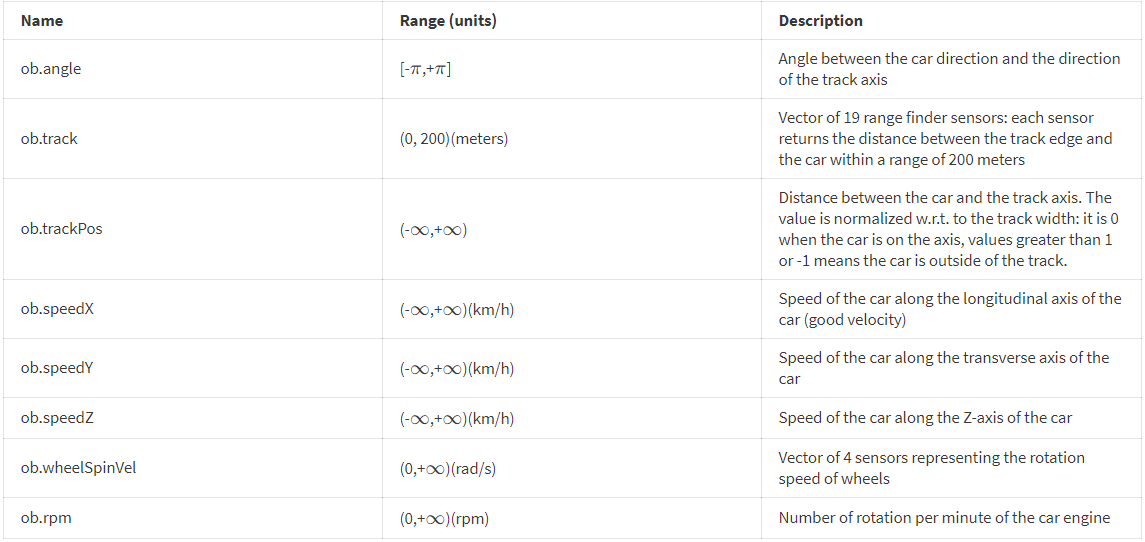
感知通常是通过传感器(摄像头，LiDAR、GPS/IMU、雷达和声呐等)获取当前环境的信息。决策是对下一个行为的控制方案（包括路由规划，行为规划等）的预测，控制则是实实在在的加多大油门，踩多大力度刹车或者是转多大角度方向等一系列对汽车的Action，而所作的这一切均与如何在道路上安全行驶有关，本文重点集中在如何进行决策。

注： 由于自动驾驶系统是一个非常复杂的工程系统，其中控制应该具备多级安全防护！因此本文仅基于模拟环境的测试对方向盘转向角，油门大小等做预测，并对该过程利用强化学习进行建模，并在游戏环境中进行模拟（让大家有点失望，毕竟实车投资有点高，平民还是很难搞），后续将会单独分享关于车辆经过十字路口等更多的一些模拟应用场景！

对于感知模块来说，摄像头一般使用障碍物检测识别、红绿灯识别等，雷达主要用于3D地图距离检测等、本文重点利用强化学习在torcs游戏上进行控制，并通过控制一个比较好的速度、刹车等多个变量之间关系。

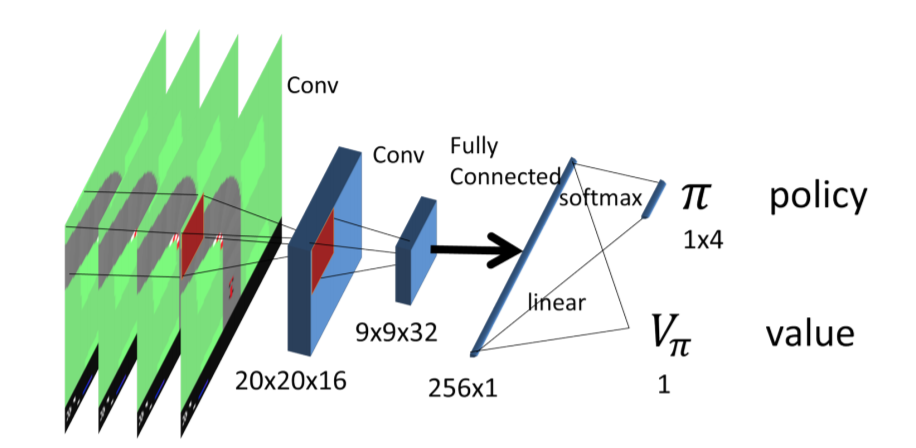
Torcs游戏简介：

《The Open Racing Car Simulator》（TORCS）是一款[开源](https://zh.wikipedia.org/wiki/%E5%BC%80%E6%BA%90)[3D](https://zh.wikipedia.org/wiki/3D)[赛车模拟游戏](https://zh.wikipedia.org/wiki/%E8%B5%9B%E8%BD%A6%E6%B8%B8%E6%88%8F)。是在[Linux](https://zh.wikipedia.org/wiki/Linux)操作系统上广受欢迎的赛车游戏。有50种车辆和20条赛道，简单的视觉效果。用[C](https://zh.wikipedia.org/wiki/C)和[C++](https://zh.wikipedia.org/wiki/C%2B%2B)写成，释放在[GPL](https://zh.wikipedia.org/wiki/GPL)协议下。本文使用的torcs游戏大概有18中输入维度，部分如下图



问题建模：

通过摄像头获取前方的场景，通过CNN卷积处理后，通过强化学习方法得到一套控制速度的策略，整体架构如图：

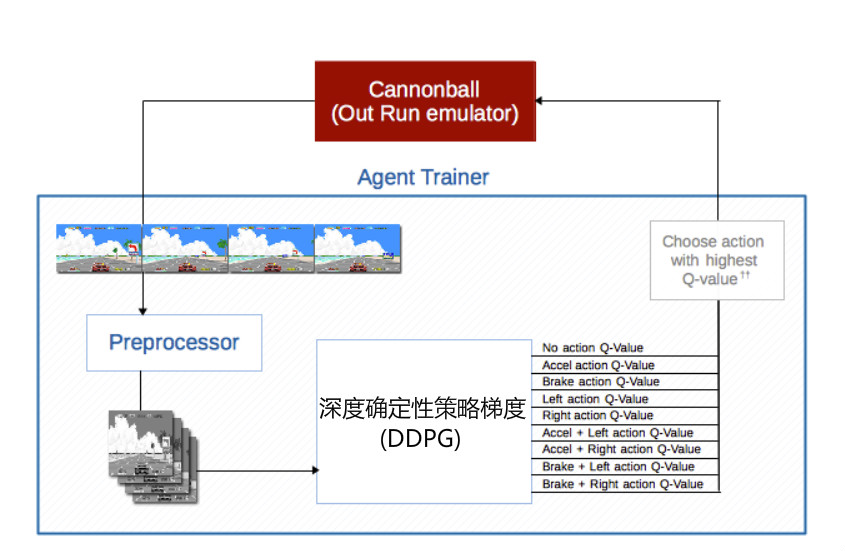


# 4、强化学习算法

前沿：本文使用基于gym平台的Torcs游戏进行模拟学，采用深度确定性策略梯度算法进行实验。为什么选择确定性策略梯度算法呢：

原因：对于游戏中的参数来说，油门，刹车板等变量都是连续性变量，像原始的DQN算法在连续性策略上表现并不是特别优秀，2015年David Sliver证明了确定性策略梯度的存在后，利用深度神经网络实现了DDPG,它在处理连续性、状态空间特别大的场景下具有优秀的表现。

本文使用的DDPG是利用了AC框架进行学习，采用了异策略方法：即actor采用了随机策略进行探索环境，而Critic网络则采用了确定性策略进行函数逼近。



## 4.1环境安装

机器使用环境：Ubuntu16.04, tensorflow1.3,python2.7

sudo apt-get install xautomation

sudo pip install numpy

sudo pip install gym

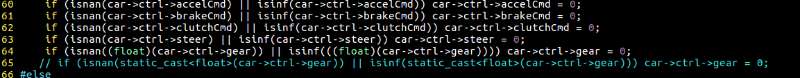
git clone <https://github.com/ugo-nama-kun/gym_torcs.git>

cd gym\_torcs/vtorcs-RL-color/src/modules/simu/simuv2/

sudo vim simu.cpp

// annotation lines-64 and add the following codes

//if(isnan((float)(car->ctrl->gear)) ||isinf(((float)(car->ctrl->gear)))) car->ctrl->gear = 0;



cd gym\_torcs/vtorcs-Rl-color目录下

// install dependcy package

sudo apt-get install libglib2.0-dev libgl1-mesa-dev libglu1-mesa-dev freeglut3-dev libplib-dev libopenal-dev libalut-dev libxi-dev libxmu-dev libxrender-dev libxrandr-dev libpng12-dev

./configure

// make enviroement

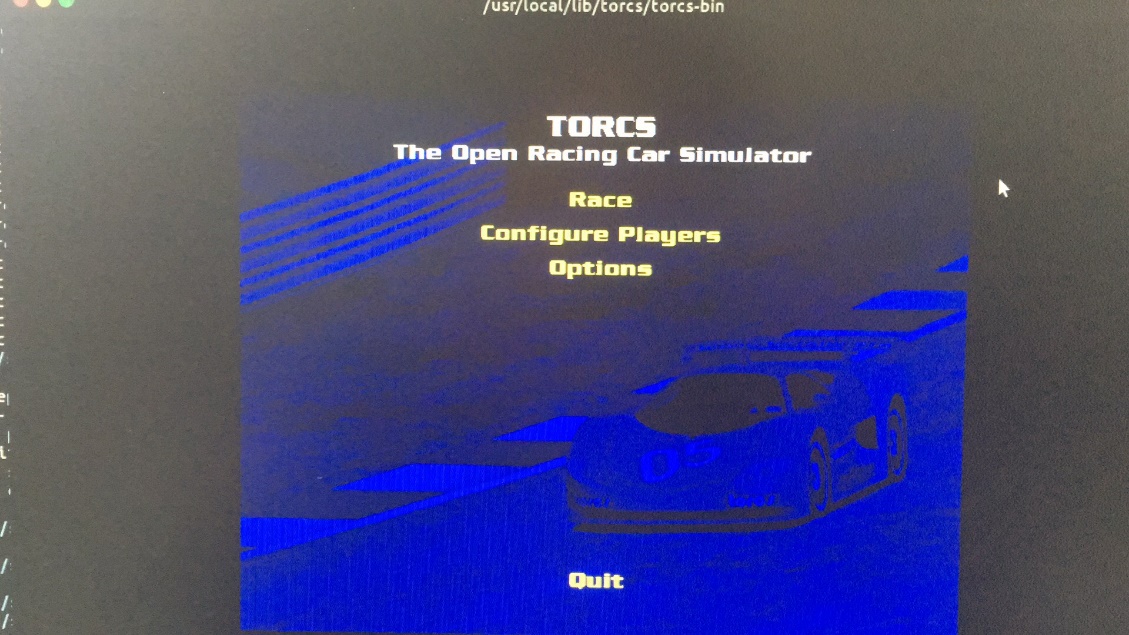
make

sudo make install

sudo make datainstall

// install finished and start games by command torcs

torcs



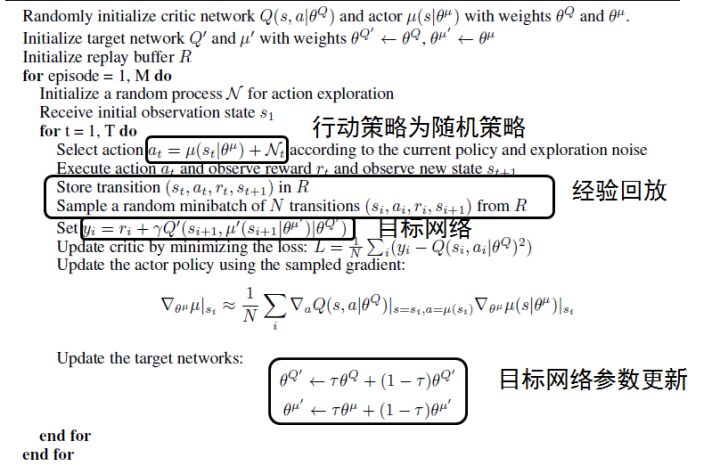
## 4.2算法讲解：深度确定性策略(DDPG)算法

DDPG算法使用了DQN算法的两个非常重要的思想：

（1）经验回放

（2）目标网络

之所以使用这两个思想，就是为了能够消除强化学习在数据采集中的相关性，其伪代码如下：



### 4.2.1确定性策略

在系列文章第一篇中已经讲过确定性策略和随机策略的概念了，重新提一下:

(1)确定性策略：

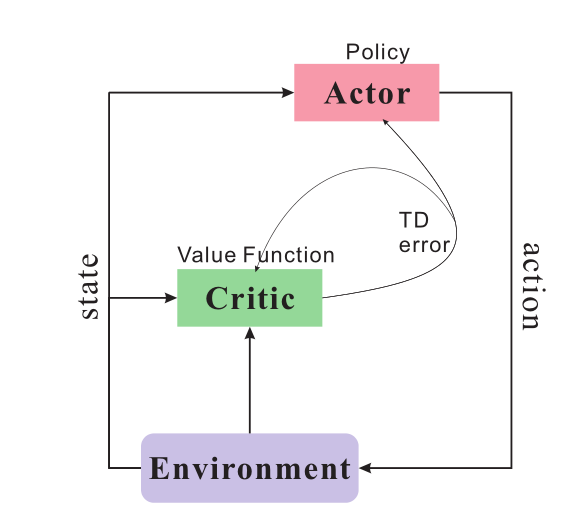
(2)随机策略: 

那什么是确定性策略呢？

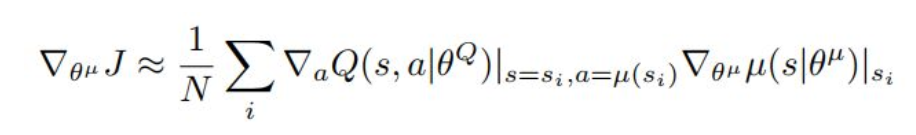
### 4.2.2表演-评论者（AC）框架

所谓的AC框架就是演员评论家（Actor-critic）框架, 就相当于一个演员在舞台上跳舞，下面的评论家高速演员他所做动作的好坏，然后演员逐渐改善自己的动作，

其实算法中采用了异策略，即Actor通过随机策略进行探索，而Critic则采用确定性策略，整体的结构如图：



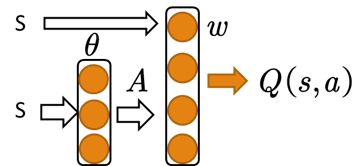
Actor的随机策略的梯度：



Critic的更新方式：

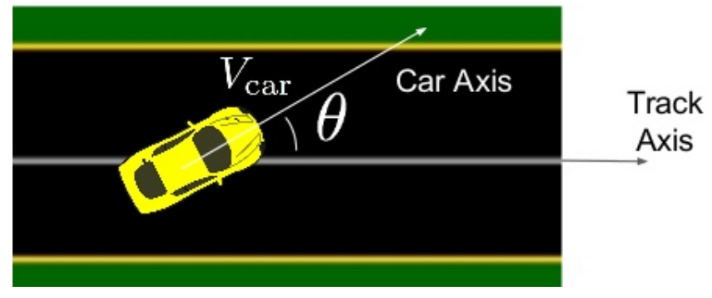


学习结构如下：



Critic根据环境状态S对Actor做出的动作进行评价！

### 4.2.3奖励函数设置：





## 4.3 代码运行与分析

### 4.3.1代码运行

程序运行环境（tf.10、keras1.1、python2.7）

pip install keras

pip install tensorflow

git clone https://github.com/yanpanlau/DDPG-Keras-Torcs.git

cd DDPG-Keras-Torcs

cp \*.\* ../gym\_torcs

cd ../gym\_torcs

python ddpg.py

效果图：



### 4.3.2代码分析

**参考文献：**

[1]. http://geek.csdn.net/news/detail/199280

[2]. http://www.fzb.me/2017-9-25-driving-by-wire-tech.html

[3]. Pedal simulator and braking feel evaluation in brake by wire system

[4]. <https://blog.csdn.net/gsww404/article/details/79630234>

[5]. M. A. Farhan Khan,Car Racing using Reinforcement Learning[J]

[6].https://lopespm.github.io/machine\_learning/2016/10/06/deep-reinforcement-learning-racing-game.html

[7]. <https://yanpanlau.github.io/2016/10/11/Torcs-Keras.html>

[8]. <https://arxiv.org/pdf/1304.1672.pdf>

[9].David sliver, Deterministic Policy Gradient Algorithms