



# 机器人局部动态避障

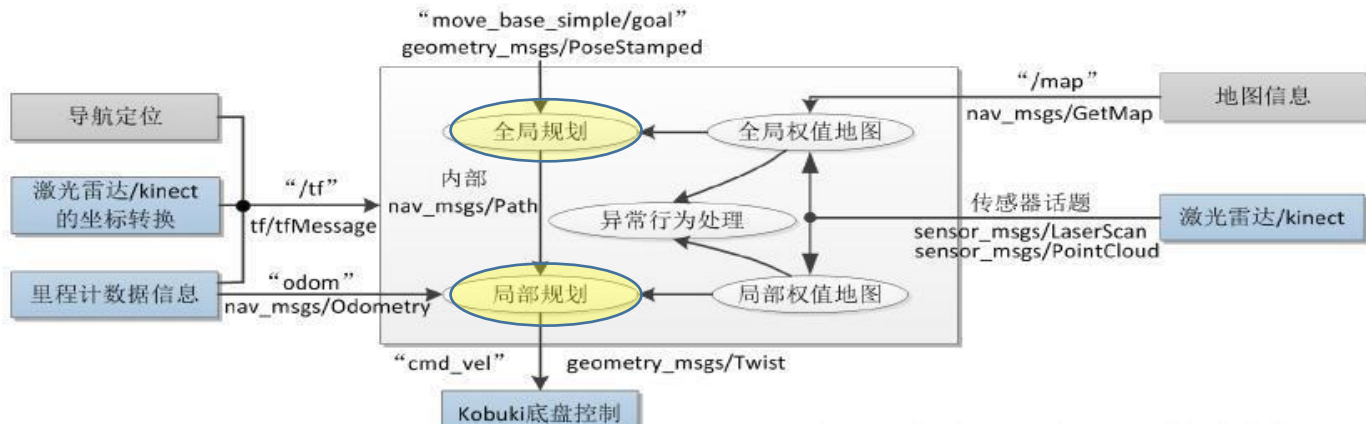
吴毅红，中国科学院大学，2020春季



# 局部路径规划

## 基本概念

机器人在获得目的地信息后，首先经过全局路径规划规划出一条大致可行的路线，然后调用局部路径规划器根据这条路线及costmap的信息规划出机器人在局部时应该做出的具体行动策略。局部路径规划在论文中通常也被称作避障（Obstacle avoidance）。



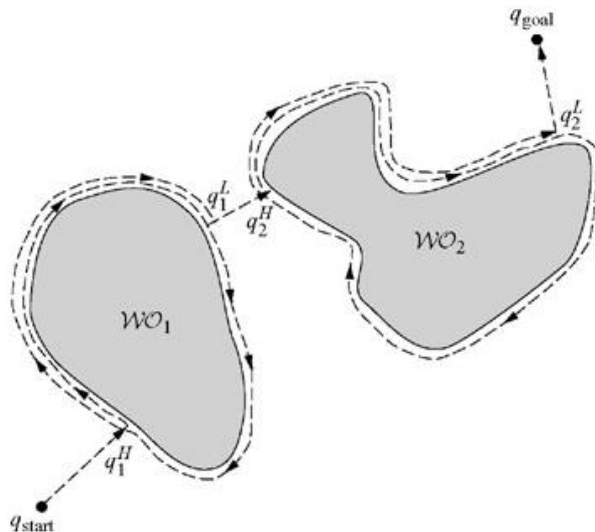
- 传统方法

- BUG算法
- 向量势直方图法（Vector Field Histogram）
- 动态窗口法（Dynamic Window Approach）
- **Velocity Obstacle**系法

- 基于深度增强学习的方法

# BUG算法

如果遇到障碍，则称点 $q_1^H$ 为第一次遇到障碍时的撞击点（*hit point*）。接着，机器人环绕障碍物移动直至返回 $q_1^H$ 点。然后判断出障碍物周边上离目标最近的点，并移到这个点上。效率很低但是肯定具有完备性。

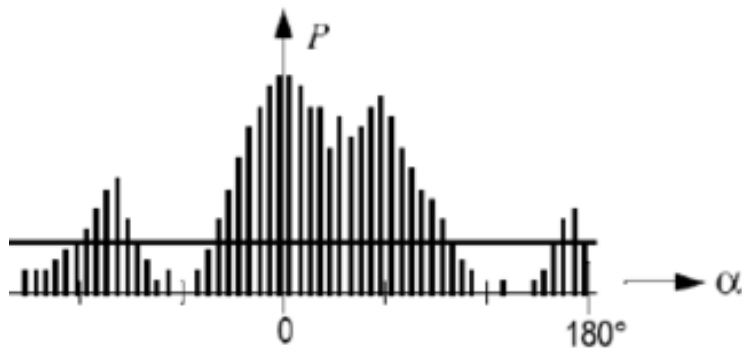


# VFH (Vector Field Histogram) 向量势直方图法

x 轴代表行走的方向与障碍物所构成的角度, y 轴是根据占有栅格的多少来计算障碍物

## • 基本步骤:

- 算法内构建并维护机器人周围环境的局部栅格地图
- 根据占用栅格地图的单元值计算障碍物概率直方图
- 根据直方图, 计算导航方向
  - 识别所有可以让机器人通过的通道
  - 对每个通道计算成本
  - 选择具有最低成本的通道



成本函数  $G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$

**target\_direction**: 路径与目标之间的对齐量

**wheel\_orientation**: 新方向和当前机器人方向的差异量

**Previous\_direction**: 原来选择方向和新方向之间的差异量

$$L = k * \theta_{tar} + u * \theta_{wheel} + w * \theta_{pre}$$

其中,  $\theta_{tar}$  表示目的角度;

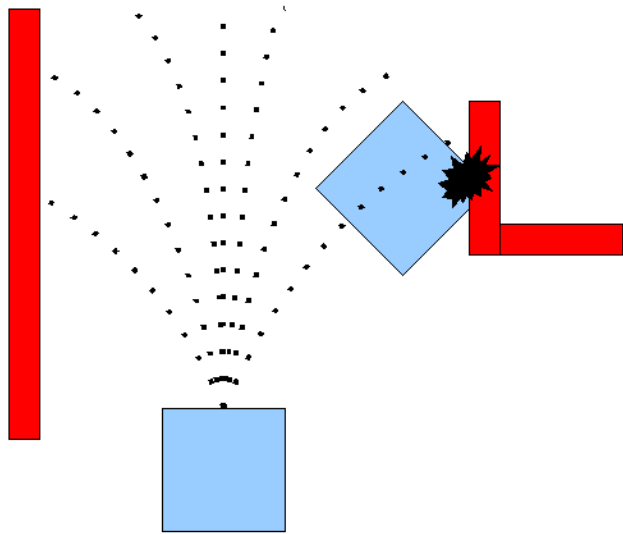
$\theta_{wheel}$  表示轮子角度;

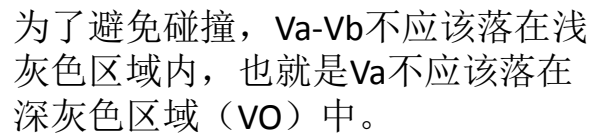
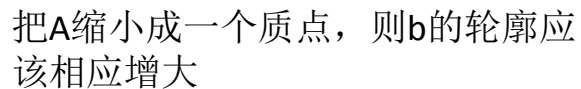
$\theta_{pre}$  表示原来角度

# DWA (Dynamic Window Approach) 动态窗口法

DWA算法基本思路如下:

- 1.在机器人控制空间进行速度离散采样(dx,dy,dtheta)
- 2.对每一个采样速度执行前向模拟,看看使用该采样速度移动一小段段时间后会发生什么
- 3.评价前向模拟中每个轨迹,评价准则如: 轨迹与障碍物距离, 轨迹是否指向目标, 轨迹是否贴近全局规划器得到的路径。挑出得分最高的轨迹并发送相应速度给移动底座
- 4.重复上面步骤.





# 基于深度增强学习的方法

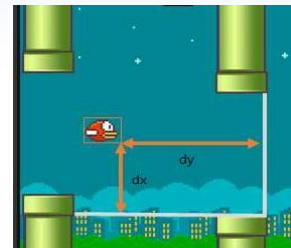
传统的路径规划方法根据机器人与行人之间的几何关系与预先设计好的规则输出下一时刻机器人的控制信息。这一类方法有时无法准确地预计出行人的轨迹，进而导致机器人与行人发生碰撞。

由于深度增强学习在AlphaGo等项目中展现的潜力较大，部分研究者开始尝试利用该技术解决路径规划问题，并且取得了较为不错的成果。经过增强学习框架的训练后，机器人能够学会人类右侧或左侧的通行规则，且算法整体计算量较小，实时性满足要求。

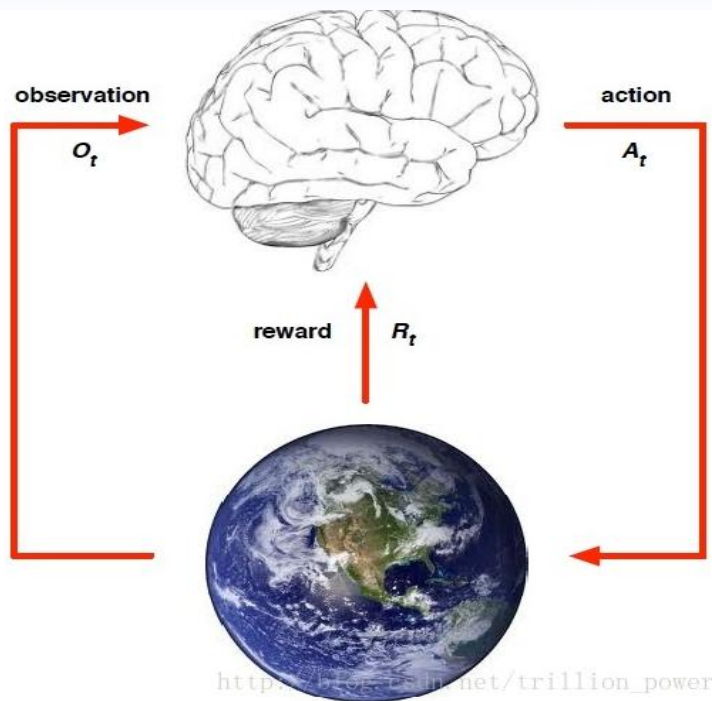






- Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop 2013.
- Chen Y F, Everett M, Liu M, et al. Socially aware motion planning with deep reinforcement learning. IEEE International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 1343-1350.
- Changan Chen, Yuejiang Liu, Sven Kreiss and Alexandre Alahi. Crowd-Robot Interaction: Crowd-aware Robot Navigation with Attention-based Deep Reinforcement Learning, ICRA 2019. (<https://github.com/vita-epfl/CrowdNav>)



# 总结1. 增强学习的几个基本概念



1. Agent 智能体
2. State 状态
3. Action 动作
4. Reward 反馈奖励

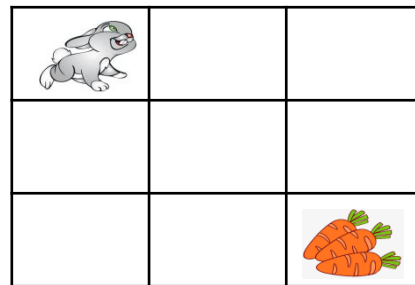
增强学习解决的是一个序列决策优化问题，就是通过不断的尝试记住各个情况下的好坏，最后找到一个最优的策略从而使Reward最多。

## 总结2. Q-learning算法:

Q-learning 算法是增强学习算法的一种。

$Q(s,a)$ :动作价值函数, 给定当前状态 $s$ 和动作 $a$ , 将获得的预期总奖励。

$Q(s,a)$ :	上	下	左	右
(1,1)				
(1,2)				
(1,3)				
(2,1)				
(2,2)				
(2,3)				
(3,1)				
(3,2)				
(3,3)				



当 $Q$ 函数已知时, 即可通过查表的方式, 执行当前状态下可获得预期总奖励最多的动作。

## 总结2. Q-learning算法:

### 如何学习 $Q(s,a)$ 函数呢?

初始化  $Q(s,a)$

For episode=1:M do

    初始化状态  $s$

    For t=1:T do

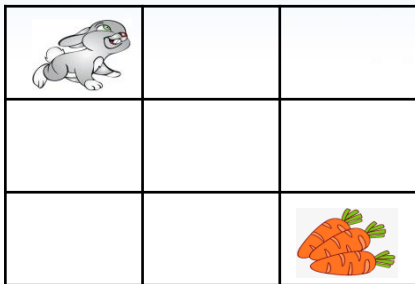
        使用  $\epsilon$ -greedy 策略选取一个动作  $a$  执行

        执行完动作, 观察 reward 和新的状态  $s_{t+1}$

        更新  $Q$  值,  $Q(s_t, a) = (1 - \alpha)Q(s_t, a) + \alpha[Reward + \gamma \max_a Q(s_{t+1}, a)]$

    End for

End for

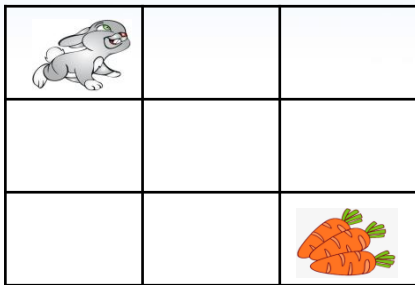


1.  $\epsilon$ -greedy 策略是一种简单的平衡“经验”和“探索”的方法。智能体以  $1 - \epsilon$  的概率根据学习到的  $Q$  函数行动, 以  $\epsilon$  的概率随机行动, 用于探索新的经验。
2.  $\alpha$  是学习率
3.  $\gamma$  是衰减系数

## 总结2. Q-learning算法:

增强学习的三个基本问题:

1. 明确增强学习的基本概念与特定项目中概念的对应关系。
2. 明确学习目标, 以及学习目标与执行动作的关系。
3. 如何学习目标。

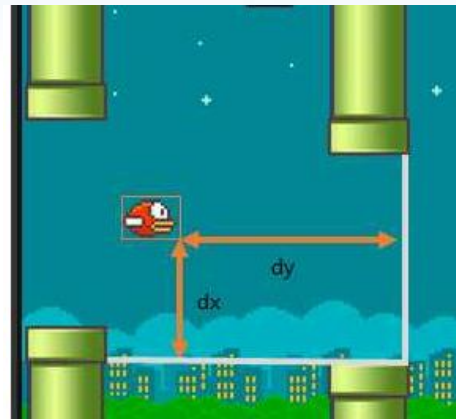


## 总结3. Playing Atari with Deep Reinforcement Learning:

问题描述：从原始游戏画面出发，通过增强学习自动学出玩游戏的方法

### 1. 明确增强学习的基本概念与特定项目中概念的对应关系。

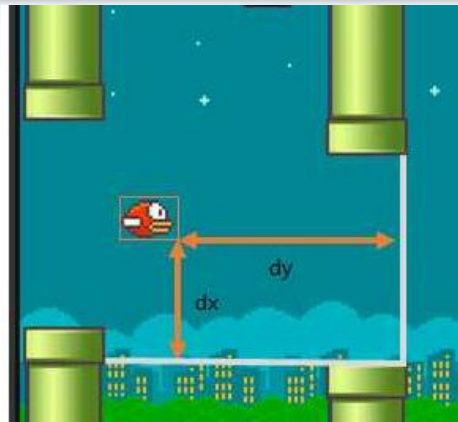
- 1). Agent智能体 → bird
- 2). State状态 → 当前游戏画面
- 3). Action动作 → 向上飞一下，不飞
- 4). Reward反馈奖励 → 小鸟活着时，每一帧给予1的奖赏；  
若死亡，则给予-1000的奖赏；  
若成功经过一个水管，则给予50的奖赏。



## 总结3. Playing Atari with Deep Reinforcement Learning:

### 1. 明确增强学习的基本概念与特定项目中概念的对应关系。

- 1). Agent智能体 → 玩家/bird
- 2). State状态 → 当前游戏画面
- 3). Action动作 → 向上飞一下，不飞
- 4). Reward反馈奖励 → 小鸟活着时，每一帧给予1的奖赏；若死亡，则给予-1000的奖赏；若成功经过一个水管，则给予50的奖赏。



### 2. 明确学习目标，以及学习目标与执行动作的关系。

学习目标：  $Q(s,a)$ ，其中  $s$  是当前游戏画面， $a$  是动作（向上飞一下，不飞）

动作执行策略：执行当前状态下可获得预期总奖励最多的动作，

$$\text{即 } \pi(s) = \max_a Q(s, a)。$$

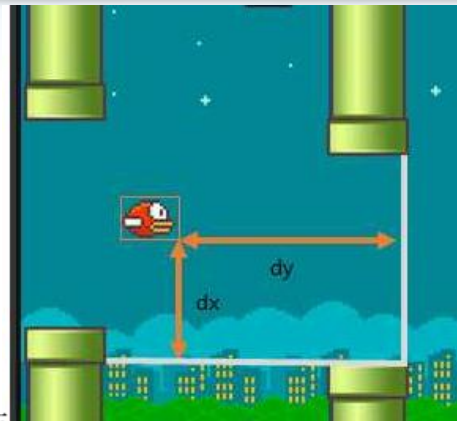
# 总结3. Playing Atari with Deep Reinforcement Learning:

## 2.明确学习目标，以及学习目标与执行动作的关系。

学习目标:  $Q(s,a)$ , 其中 $s$ 是当前游戏画面,  $a$ 是动作

动作执行策略: 执行当前状态下可获得预期总奖励最多的动作,

$$\pi(s) = \max_a Q(s,a)。$$



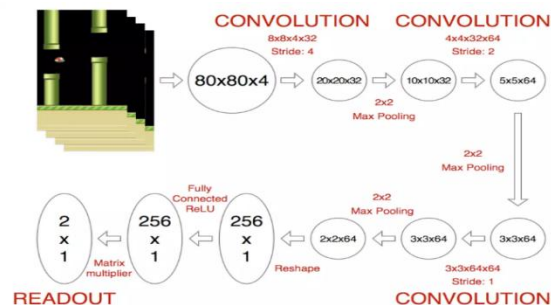
**问题:** 在游戏中, 再用“表格”来表示 $Q$ 函数就不合适了。原因在于Atari游戏画面为 $210 \times 160$ 的RGB图像, 假设用一帧图像表示一个状态, 就算每个像素位置只有0和1来表示, 产生的状态也高达 $2^{(210 \times 160)}$ 种, 这意味着 $Q$ 函数对应的表格有 $2^{(210 \times 160)}$ 行, 根本无法存储下来, 也无法进行训练。

**解决方法:** 深度学习。

网络输入: 游戏画面

网络输出: 二维向量,

分别表示 $Q(s, \text{飞})$ 和 $Q(s, \text{不飞})$





## 总结3. Playing Atari with Deep Reinforcement Learning:

### 3.如何学习目标。

#### Algorithm 1 Deep Q-learning with Experience Replay

```

Initialize replay memory  $\mathcal{D}$  to capacity  $N$ 
Initialize action-value function  $Q$  with random weights
for episode = 1,  $M$  do
  Initialise sequence  $s_1 = \{x_1\}$  and preprocessed sequenced  $\phi_1 = \phi(s_1)$ 
  for  $t = 1, T$  do
    With probability  $\epsilon$  select a random action  $a_t$ 
    otherwise select  $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 
    Execute action  $a_t$  in emulator and observe reward  $r_t$  and image  $x_{t+1}$ 
    Set  $s_{t+1} = s_t, a_t, x_{t+1}$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
    Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in  $\mathcal{D}$ 
    Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from  $\mathcal{D}$ 
    Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
    Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
  end for
end for
  
```

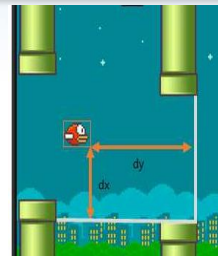
初始化网络和经验池

使用 $\epsilon$ -greedy策略选取一个动作 $a$ 执行

执行完动作，观察reward和新的状态 $s_{t+1}$

加入经验池。

从经验池中随机取出样本，利用误差反传训练网络。



经验回放打破了数据间的关联性，让训练收敛且稳定。

## 总结4.

# Socially aware motion planning with deep reinforcement learning :

问题描述：从当前位置出发，规划躲避行人并到达目的地的路径。



## 1. 明确增强学习的基本概念与特定项目中概念的对应关系。

- 1). Agent智能体 → 机器人
- 2). State状态 → 智能体的位置、速度、半径等
- 3). Action动作 → 速度（向量）
- 4). Reward反馈奖励 → 根据几何信息和右手或左手准则设计

## 总结4.

### Socially aware motion planning with deep reinforcement learning :

#### 1. 明确增强学习的基本概念与特定项目中概念的对应关系。

- 1). Agent智能体 → 机器人
- 2). State状态 → 智能体的位置、速度、半径等
- 3). Action动作 → 速度（向量）
- 4). Reward反馈奖励 → 根据几何信息和右手或左手准则设计



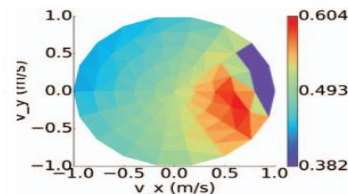
#### 2. 明确学习目标，以及学习目标与执行动作的关系。

学习目标:  $V(s)$ , 状态价值函数, 智能体从状态 $s$ 开始将获得的预期总奖励。由于机器人动作空间不具有离散性, 因此采用状态值函数 $V(s)$ , 而不是 $Q(s,a)$ 。

动作执行策略: 执行当前状态下可获得预期总奖励最多的动作,

$$U \leftarrow \text{sampleActions}(\quad),$$

$$u_t = \operatorname{argmax}_{u \in U} \{R(s_t, u_t) + \gamma V(s_{t+1})\}, \text{ where } s_{t+1} \leftarrow \text{prog}(s_t, \Delta t, u_t).$$



## 总结4.

# Socially aware motion planning with deep reinforcement learning :

## 3.如何学习目标。



### Algorithm 1: Deep V-learning for SA-CADRL

```

1 initialize and duplicate a value net with  $n$  agents
   $V(\cdot; \theta, n), V' \leftarrow V$ 
2 initialize experience sets  $E \leftarrow \emptyset, E_b \leftarrow \emptyset$ 
3 for  $episode=1, \dots, N_{eps}$  do
4   for  $m$  times do
5      $p \sim \text{Uniform}(2, n)$ 
6      $s_0^1, s_0^2, \dots, s_0^p \leftarrow \text{randomTestcase}(p)$ 
7      $s_{0:t_f}^1, s_{0:t_f}^2, \dots, s_{0:t_f}^p \leftarrow \text{SA-CADRL}(V)$ 
8     with prob  $\epsilon_f$ , mirror every traj  $s_{0:t_f}^i$  in the x-axis
9     for every agent  $i$  do
10       $y_{0:T}^i \leftarrow \text{findValues}(V', s_{0:t_f}^{j^n, i})$ 
11       $E, E_b \leftarrow \text{assimilate}(E, E_b, (y^i, s^{j^n, i})_{0:t_f})$ 
12    $e \leftarrow \text{randSubset}(E) \cup \text{randSubset}(E_b)$ 
13    $\theta \leftarrow \text{RMSprop}(e)$ 
14   for every  $C$  episodes do
15     Evaluate( $V$ ),  $V' \leftarrow V$ 
16 return  $V$ 

```

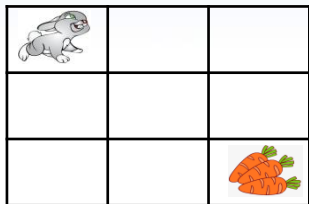
初始化网络和经验池

随机初始化智能体的位置  
使用 $\epsilon$ -greedy策略规划一个路径，即一个从初始位置到终点位置的位置序列。

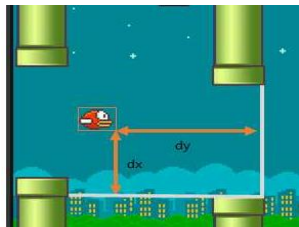
找到每条路径对应的状态值，  
组成state-value pairs，加入经验池。

从经验池中随机取出state-value样本，  
利用误差反传训练网络。

# 总结 all



1. 学习目标：动作价值函数，即 $Q(s,a)$ 函数。
2. 如何使用学习好的 $Q$ 函数：可通过查表的方式，执行当前状态下可获得预期总奖励最多的动作。



1. 学习目标： $Q(s,a)$ 函数。
2. 主要改进：由于状态数量庞大，引入深度学习。
3. 深度网络输入：游戏画面；输出：每个动作对应的 $Q$ 值。
4. 如何使用训练好的 $Q$ 网络：输入当前游戏画面，输出每个动作 $Q$ 值，执行当前状态下可获得预期总奖励最多的动作。



1. 学习目标：状态值函数 $V(s)$ 。
2. 主要改变：由于机器人动作空间不具有离散性，因此采用状态值函数 $V(s)$ ，而不是 $Q(s,a)$ 。
3. 深度网络输入： $s$ ；输出： $V(s)$ 。
4. 如何使用训练好的 $V$ 网络：输入状态 $s$ ，根据训练网络估计采样动作下的预期总奖励，执行当前状态下可获得预期总奖励最多的动作：

$$U \leftarrow \text{sampleActions}(), \\ u_t = \text{argmax}_{u \in U} \{R(s_t, u_t) + \gamma V(s_{t+1})\}$$

# 课后作业

- 了解SpaceX的龙飞船卫星发射新闻
- SpaceX公司火箭精准着陆回收，用到的有可能是什么导航定位技术？
- 畅想SpaceX的“星链”计划对未来定位导航的影响

# 谢 谢

yhwu@nlpr.ia.ac.cn

