A Reinforcement Learning Method of Obstacle Avoidance for Industrial Mobile Vehicles in Unknown Environments Using Neural Network

Chen Xia, A. El Kamel

Gabriel V. de la Cruz Jr.

CptS 595: AI Seminar

Problem

- How to use *reinforcement learning* for autonomous navigation of industrial mobile vehicle or robots in *unknown environment*?

Motivation

- Mobile robots are interesting for its vast potential uses:
  - Warehouse transportation & distribution
  - Use in hazardous applications
  - Home care
  - Autonomous cars
  - and lots more…

Warehouse Transportation & Distribution

https://www.youtube.com/watch?v=lWsMdN7HMuA

Warehouse Transportation & Distribution

https://www.youtube.com/watch?v=G_pbiDf3i30

Motivation

- Safety
  - Humans
  - Robots
- **Adaptability**
  - Classic path planning vs RL

Alternatives

- Global path planning
- Local path planning techniques
  - Fuzzy Logic Control [Raguranan et al]
  - Potential Field Method [Ge & Cui]
  - Genetic Algorithms [Hu & Yang]

Fuzzy Logic Control (Raguranan et al)

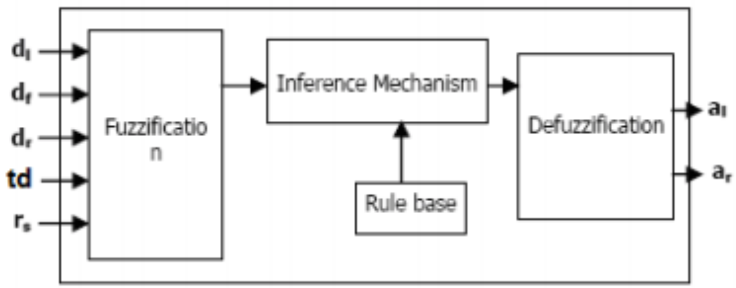


Fig. 2. Fuzzy logic controller

IF
Obstacle distance on left (d_l) is Near and
Obstacle distance on front (d_f) is Far and
Obstacle distance on right (d_r) is Far and
Target direction (t_d) is Right and
Current robot speed (r_s) is Slow

THEN
Acceleration of left wheel (a_l) is PB
Acceleration of right wheel (a_r) is PS

Fig. 4. Example of the inference rules. d_l = NEAR, {FAR}, d_f = {NEAR, FAR}, d_r = {NEAR, FAR}, t_d = {LEFT, CENTER, RIGHT}, r_s = {SLOW, FAST}, a_l = {PB, PS, Z, NS, NB}, a_r = {PB, PS, Z, NS, NB}.

TABLE 1
RULE BASE FOR TARGET SEEKING BEHAVIOR

Rule no	Input					Output	
	d_l	d_f	d_r	t_d	r_s	a_l	a_r
1	F	F	F	L	SL	PS	PB
2	F	F	F	L	FS	NS	Z
3	F	F	F	C	SL	PB	PB
4	F	F	F	C	FS	Z	Z
5	F	F	F	R	SL	PB	PS
6	F	F	F	R	FS	Z	NS
7	F	F	N	L	SL	Z	PS
8	F	F	N	L	FS	NS	Z
9	N	F	N	C	SL	PS	PS
10	N	F	N	C	FS	NS	NS
11	N	F	F	R	SL	PS	Z
12	N	F	F	R	FS	Z	NS

TABLE 2
RULE BASE FOR OBSTACLE AVOIDANCE BEHAVIOR

Rule no	Input					Output	
	d_l	d_f	d_r	t_d	r_s	a_l	a_r
1	F	F	N	C	SL	Z	PS
2	F	F	N	C	FS	NS	Z
3	F	F	N	R	SL	NS	Z
4	F	F	N	R	FS	NB	NS
5	F	N	N	L	SL	NS	Z
6	F	N	N	L	FS	NB	NS
7	F	N	N	C	SL	NS	Z
8	F	N	N	C	FS	NB	NS
9	F	N	N	R	SL	NS	Z
10	F	N	N	R	FS	NB	NS
11	N	N	N	L	SL	NS	Z
12	N	N	N	L	FS	NB	NS
13	N	N	N	C	SL	NS	Z
14	N	N	N	C	FS	NB	NS
15	N	N	N	R	SL	Z	NS
16	N	N	N	R	FS	NS	NB
17	N	F	N	L	SL	Z	Z
18	N	F	N	L	FS	NS	NS
19	N	F	N	R	SL	Z	Z
20	N	F	N	R	FS	NS	NS
21	N	F	F	L	SL	Z	NS
22	N	F	F	L	FS	NS	NB
23	N	F	F	C	SL	PS	Z
24	N	F	F	C	FS	Z	NS
25	N	N	F	L	SL	Z	NS
26	N	N	F	L	FS	NS	NB
27	N	N	F	C	SL	Z	NB
28	N	N	F	C	FS	NS	NB
29	N	N	F	R	SL	Z	NS
30	N	N	F	R	FS	NS	NB
31	F	N	F	L	SL	NS	Z
32	F	N	F	L	FS	NB	NS
33	F	N	F	C	SL	NS	Z
34	F	N	F	C	FS	NB	NS
35	F	N	F	R	SL	Z	NS
36	F	N	F	R	FS	NS	NB

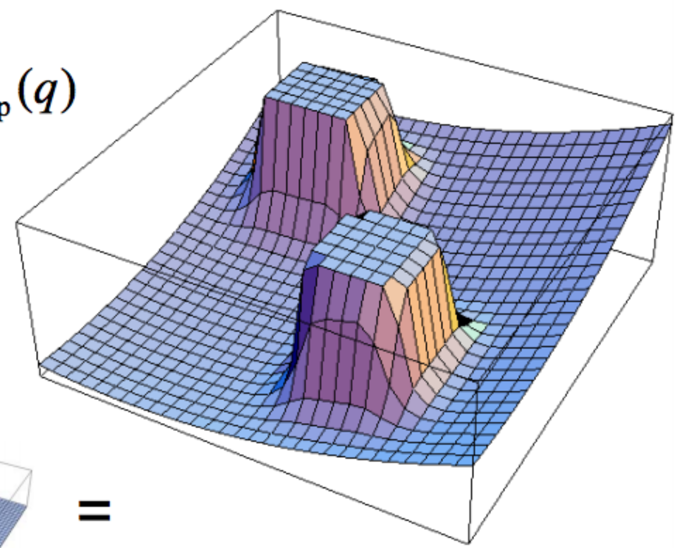
N-Near, F-Far, L-Left, C-Center, R-Right, SL-Slow, FS-Fast

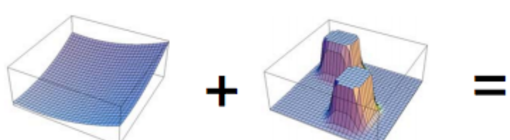
Potential Field Method [Ge & Cui]

- The basic concept is to fill the robot's workspace with an artificial potential field in which the robot is attracted to its target position and is repulsed away from the obstacles.

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

$$F(q) = -\nabla U(q)$$





Genetic Algorithms [Hu & Yang]

- Problem-specific genetic operators are designed with domain knowledge.

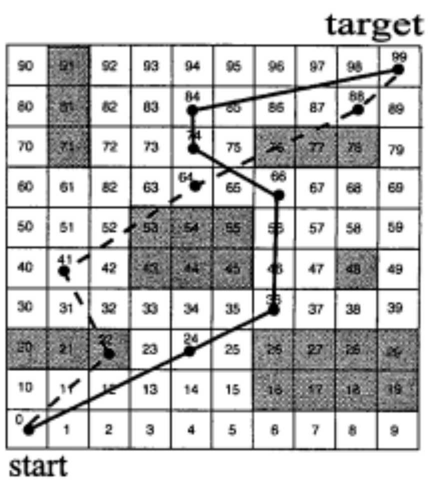


Fig. 1. Mobile robot environment and path representation. Solid line: a feasible path; dashed line: an infeasible path.

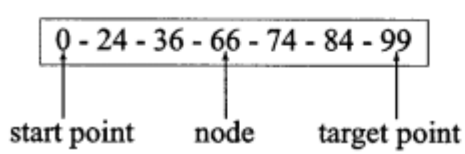


Fig. 2. A sample chromosome: a path represented by nodes falling on grids with different numbers.

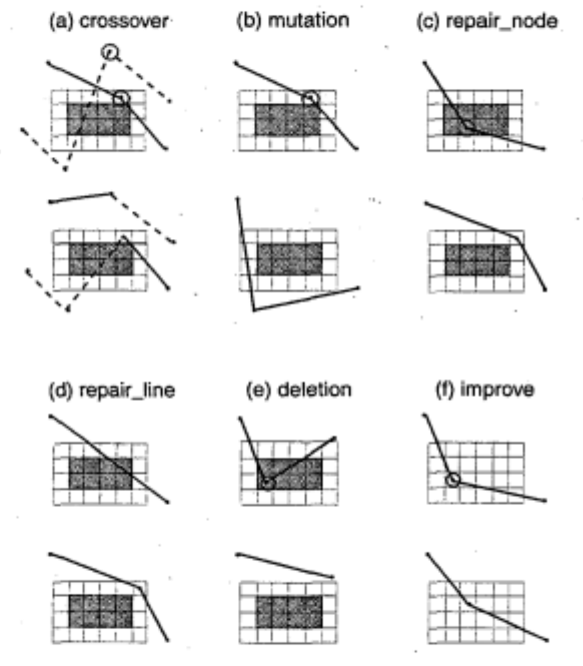
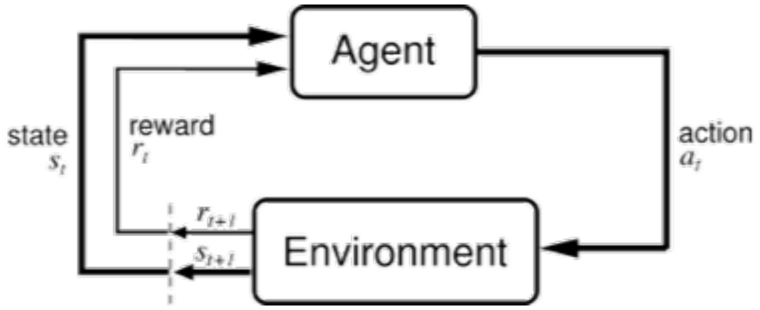


Fig. 3. Six specialized genetic operators that incorporate problem-specific knowledge

Reinforcement Learning

- It learns about the environment via interacting with it.



Neural Network based Q-Learning

Q-Learning

- RL method used for self-learning ability
- Model-free
- Off-policy
- *state-action Q values are traditionally stored in Q-table*

Neural Network

- Strong ability to deal with large-scale state spaces
- Use as function approximator

State-Action Spaces

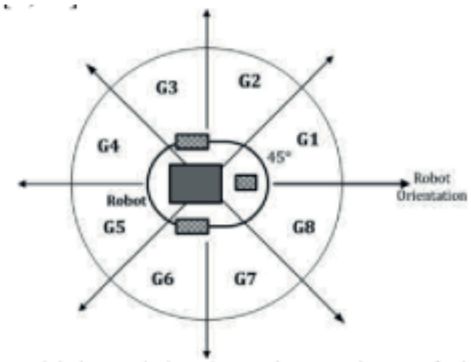


Fig.2. The vehicle and the detection regions of eight sensors

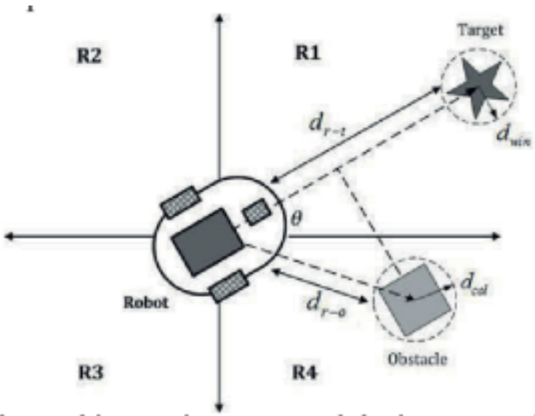


Fig.3. The working environment and the important distances

$$s_t = [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8]$$

$$actions = [Forward, Left(30^\circ), Right(30^\circ), Left(60^\circ), Right(60^\circ)]$$

Reward Function

Non-Safe State (NS) to Safe State (SS)	$r = 0.3$
Safe State (SS) to Non-Safe State (NS)	$r = -0.2$
Non-Safe State (NS) to Non-Safe State (NS) but getting closer to the obstacles	$r = -0.4$
Non-Safe State (NS) to Non-Safe State (NS) and getting away from the obstacles	$r = 0.4$
Winning State (WS)	$r = 1$
Failure State (FS)	$r = -0.6$

Q-value Function

$$Q^*(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a_i \epsilon A} Q(s_{t+1}, a_i) - Q(s_t, a_t)]$$

Action Selection

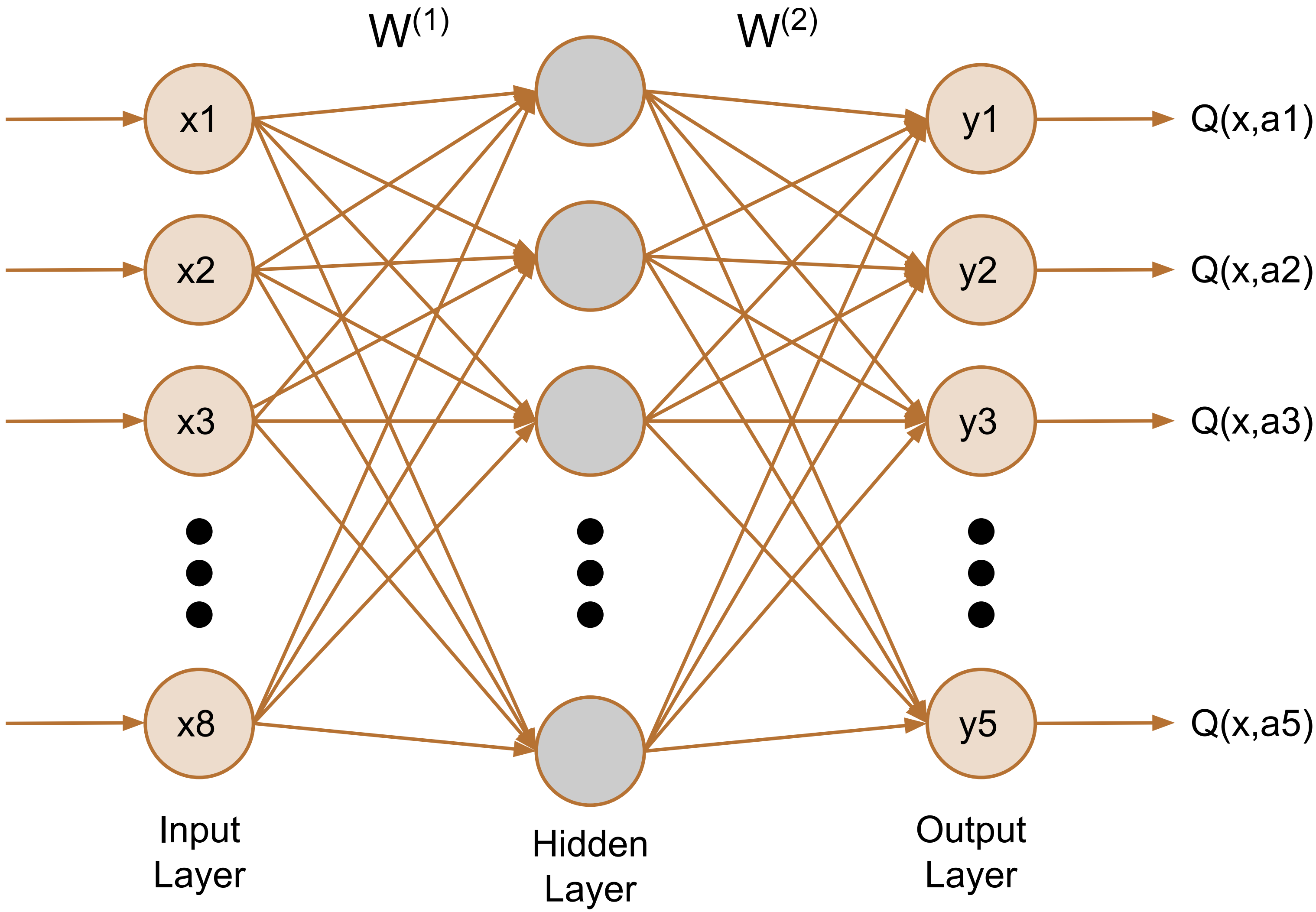
- Boltzmann probability distribution

$$P(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{b \in A} e^{Q(s,b)/T}}$$

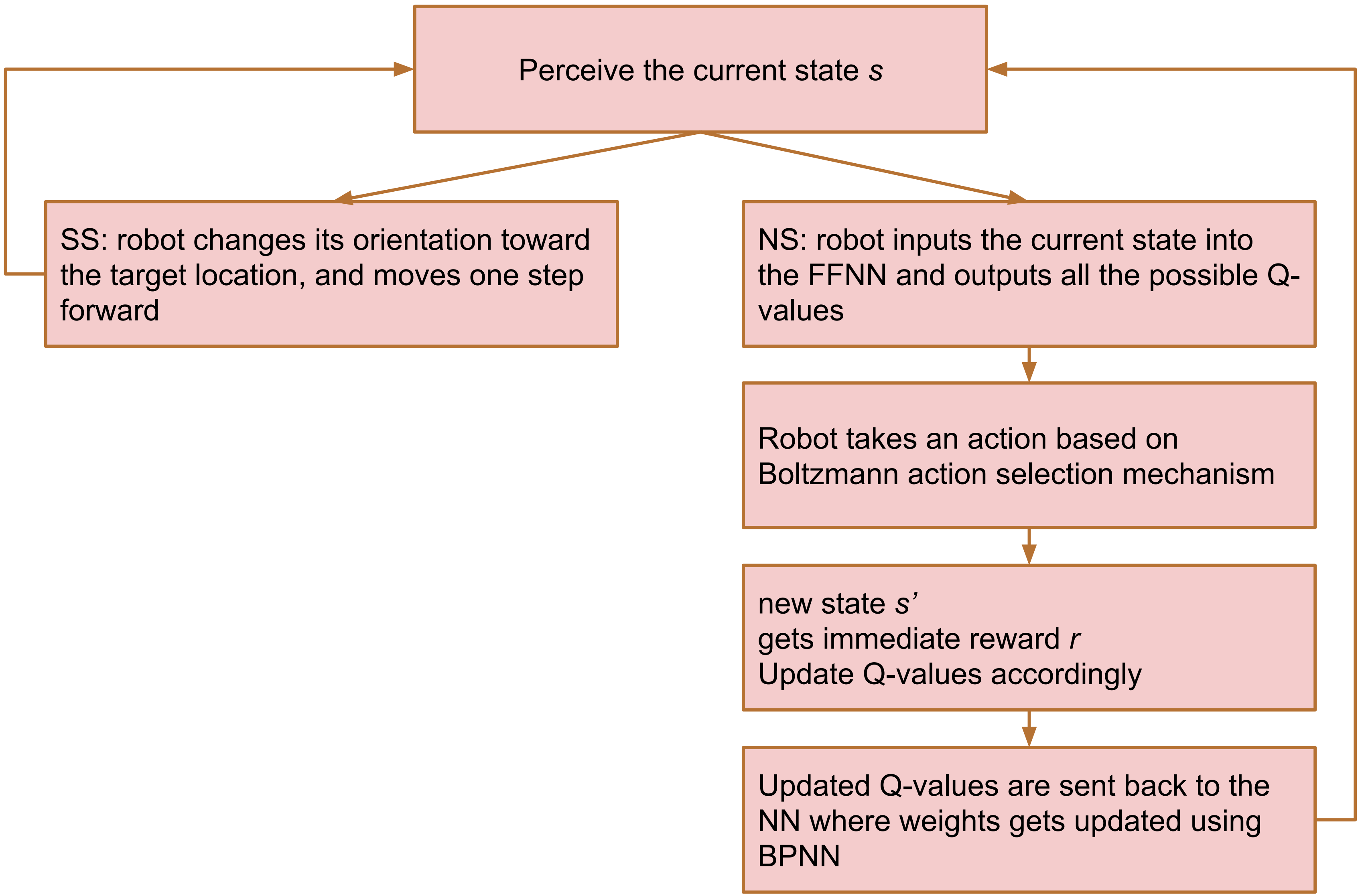
- Greedy action selection

$$a^*(s) = \arg\max_{b \in A} Q(s,b)$$

Neural Network



Algorithm


Perceive the current state *s*
SS: robot changes its orientation toward the target location, and moves one step forward
NS: robot inputs the current state into the FFNN and outputs all the possible Q-values
Robot takes an action based on Boltzmann action selection mechanism
new state *s'*
gets immediate reward *r*
Update Q-values accordingly
Updated Q-values are sent back to the NN where weights gets updated using BPNN

Simulation

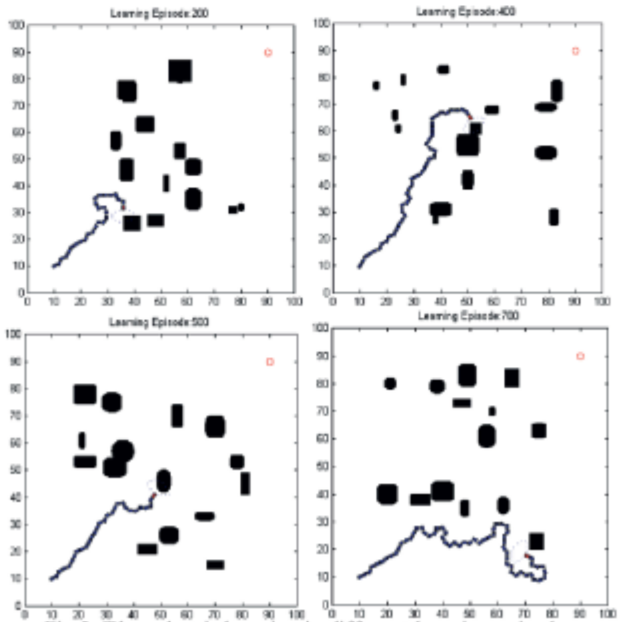


Fig.5. The robot is learning in different learning episodes

Results & Conclusion

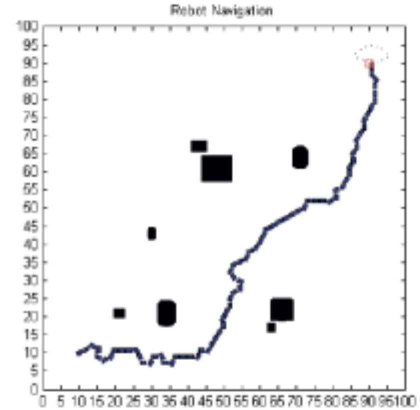


Fig.6. The robot navigation process

Then, in order to test the stability of the proposed method, the robot executed other navigation missions using the same weights and the policy, as shown in Fig.7.

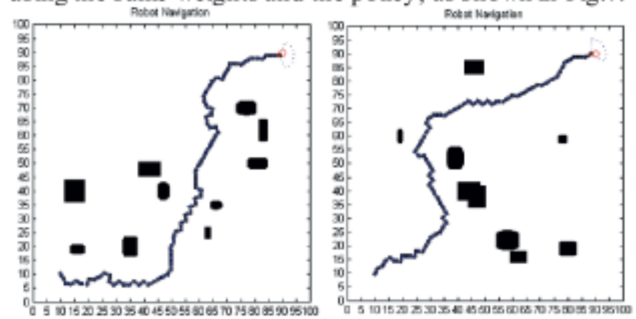


Fig.7. Other navigation processes

Deep Q-Learning

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 ϵ 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

http://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Discussion

- How relevant are neural network still with emergence of Deep Neural Networks? Should we continue using or is time to move on?
- When to use the classic Neural Network over Deep Neural Network?