Departamento de Ingeniería de Sistemas y Automática

Reinforcement Learning in the Lego® Mindstorms® NXT Robot

Ángel Martínez-Tenor

Practical Activity – Cognitive Robotics

Master Degree in Mechatronics Engineering

2015-2016

University of Málaga. System Engineering & Automation

V1.1 January 4, 2016

Introduction

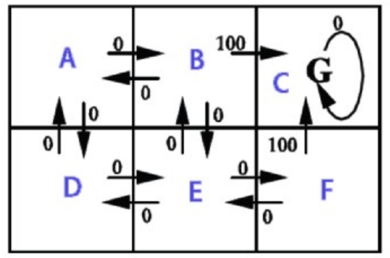
Robotics: Common subject in Engineering degrees

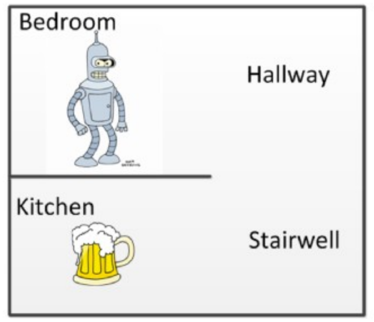
↳ **Cognitive Robotics:** Now covered in Postgraduate programs

↳ **Reinforcement Learning (RL):** Decision-Making Machine learning

↳ **Q-learning algorithm:** Simple, effective and well-known RL algorithm

Simulated robots





Embodied agent

- Obstacle avoidance
- Line follower
- Walking
- Phototactic behaviour

Agent **Environment**

action

reward

new state

RL does not require a model of the environment, overcoming this limitation by making observations

PROPOSAL:

Use a framework integrating:

- Q-learning
- Real Mobile Robots

Allowing a better understanding of the robotic learning problem

UNIVERSIDAD DE MÁLAGA

Departamento de Ingeniería de Sistemas y Automática

Developed Work

Offline Design & Simulation

- Assembly
- Modeling
- Simulation
- Parameters Tuning

→ Offline Q-learning Template Octave / Matlab

Robot implementation

- Q-learning on robot
- CPU limitations studies: Overflow & Precision losses
- Robot vs Simulation analysis
- Parameters Optimization

→ NXT Robot Q-learning NXC (main & NXT_io library)

LEGO MINDSTORMS NXT Education Base Set

PRACTICE ACTIVITY TEMPLATES

- ROBOT NXT + NXC
- ROBOT NXT + JAVA (LEJOS)
- SIM ROBOT in V-REP + Lua, Python, JAVA, C++, Matlab/Octave

Ángel Martínez-Tenor. Reinforcement Learning on the Lego Mindstorms NXT Robot. Analysis and Implementation. Master Thesis, E.T.S. de Ingenieros Industriales. Universidad de Málaga, 2013. http://babel.isa.uma.es/angelmt/Reinforcement_Learning/

Ángel Martínez-Tenor, Juan-Antonio Fernández-Madrigal, and Ana Cruz-Martín. Lego Mindstorms NXT and Q-learning: a teaching approach for robotics and engineering. In 7th International Conference of Education, Research and Innovation (ICERI, 2014), nov 2014.

Implementation: Q-learning vs SARSA

Q-learning

```
for step = 1:N_STEPS

    a = Exploitation_exploration_strategy()

    robot_execute_action(a), wait(STEP_TIME)

    sp = observe_state(), R = obtain_Reward()

    Q(s,a) = (1-alpha)*Q(s,a) + alpha*(R+GAMMA*V(sp))

    V(s)_update()
    Policy(s)_update()
    alpha_update()
    s = sp

end
```

SARSA

```
for step = 1:N_STEPS

    robot_execute_action(a), wait(STEP_TIME)

    sp = observe_state(), R = obtain_Reward()

    ap = Exploitation_exploration_strategy()

    Q(s,a) = (1-alpha)*Q(s,a) + alpha*(R+GAMMA*Q(sp,ap))

    V(s)_update()
    Policy(s)_update()
    alpha_update()
    s = sp
    a = ap

end
```

NXC Template

learning.nxc

```
long getMemoryNeeded(byte nstates, byte nactions, long nsteps);

void NXTsaveData(long memoryNeeded);

void exploitPolicy(byte s);

byte selectAction(byte s, byte strategy);

task main(){ ...

        for(step=1; step<N_STEPS+1; step++) {

            executeAction(a);
            Wait(STEP_TIME);
            sp = observeState();

            R = obtainReward(s,a,sp);

            strategy = selectActionStrategy();
            ap = selectAction(sp, strategy);

             // Update Q-matrix
            Q[s][a]=(1-alpha)*Q[s][a]  +  alpha*(R+ GAMMA*Q[sp][ap]);   SARSA

            // Update V and Policy
            { ... }

            // Update state
            s = sp;
            a = ap;
        }

... }
```

TASK.h

```
/* Constants definition --------------------------------------------------------*/
    // FileSystem parameters
    #define NAME                "Simple_1"  // LIMIT: 15 ALPHANUMERIC CHARACTERS  !!
                                            // resulting filename: NAME+".log"
    #define ENVIRONMENT         "square_70x100 (with obstacle 29x9)"

    // Q-learning algorithm parameters
    #define N_STATES            4               // from 1 to N_STATES
    #define N_ACTIONS           4               // from 1 to N_ACTIONS

    #define GAMMA               0.9     //      NXT OPTIMAL
    #define INITIAL_ALPHA       0.02    //      NXT OPTIMAL

    // Experiment parameters
    #define STEP_TIME           250             // (ms)
    #define N_STEPS             1000            // Exp_Time > STEP_TIME * N_STEPS
    #define INITIAL_POLICY      1               // 1: stop

    #define MOTOR_POWER             50          // Motor power (from 0 to 100)  TUNE
    #define THRESHOLD_DEGREES       25          // Used in reward function      TUNE
    #define THRESHOLD_DISTANCE      25          // Used in state encoding       TUNE


/* Functions prototypes --------------------------------------------------------*/

byte selectActionStrategy(void);
    /* OUTPUT    0: Greedy (exploitation)
     *           1: Random
     *           2: Least Explored
     * Note:     Refer to selectAction(byte s, byte strategy) in 'Q_LEARNING.nxc'
     *           for further details
     */

void executeAction(byte a);
  /* INPUT:     Selected action (Do not forget to check N_ACTIONS) */

byte observeState(void);
    /*  Returns the state of the robot by encoding the information measured from
     *   its sensors (ultraSonic, contacts ...). In case the number of states or
     *   their definitions change, this function  must be updated
     *  OUTPUT: Observed State (sp)
     */

float obtainReward(byte s, byte a, byte sp);
    /*  Gets and Returns the obtained reward
     *  INPUT:      Previous State (s), Action executed (a), State Reached (s')
     *  OUTPUT:     Reward
     */
```

UNIVERSIDAD DE MÁLAGA

Departamento de Ingeniería de Sistemas y Automática

NXC Template

NXT_io.h

```
#define LEFT_WHEEL        OUT_C
#define RIGHT_WHEEL       OUT_B
#define ULTRASONIC        S4
#define LEFT_BUMPER       SENSOR_3
#define RIGHT_BUMPER      SENSOR_2

void NXT_mapSensors(void);

byte getObservationUltrasonic(void);

void executeNXT(string command, byte power = DEFAULT_POWER);

void display(const string &m);
void displayForceRow(const string &m, byte row);


bool NXTcheckMemory(long memoryNeeded);
void pauseNXT(void);

bool debugButton(void);
bool exploitPolicyButton(void);
bool saveAndExitButton(void);
byte getButtonPressed(void);

void playStartSound(void);
void playEndingSound(void);
void playErrorSound(void);
void playExploitationSound(void);
void playStepSound(void);
void playStepOptimalSound(void);
void playPauseSound(void);
void playDebugSound(void);
```

SETUP (NXT_learning_setup.png)

Ultrasonic Restrictions:

- Distance < ~142 cm
- Angle between normal of obstacle and sonar < ~48 deg.

NXC Template: Output

(Q_learning.nxc)

NXT Display (online)

```
Task Name
step:
s:
a:
sp:
Reward:
Policy:[s]
```

Useful in Debug Mode

Logfile (offline)

```
% Simple_1 % Learning Algorithm: SARSA
% square_70x100 (with obstacle 29x9)
 % --------------- INPUT PARAMETERS ----------
N_STATES = 4;
N_ACTIONS = 4;
GAMMA = 0.9;
INITIAL_ALPHA = 0.02;
STEP_TIME = 250;
INITIAL_POLICY = 1;
% -------Physical ----
MOTOR_POWER = 50;
THRESHOLD_DISTANCE = 25;
THRESHOLD_DEGREES  = 25;
 % ------------------- RESULTS ---------------
steps = 1001;
Policy = [4 3 2 3 ]; % [s]
V = [7.3283 2.6846 3.1348 0.1157 ]; % [s]
Q = [
3.2313 3.5949 2.9638 7.3283
-0.8531 0.1223 2.6846 -1.7748
-0.3626 3.1348 0.0756 -0.2705
-5.7099 -0.3151 0.1157 -4.9091
]; % [s,a]
exploration = [
71 70 220 435
10 9 57 9
7 68 6 3
6 13 11 5
]; % [s,a]
```

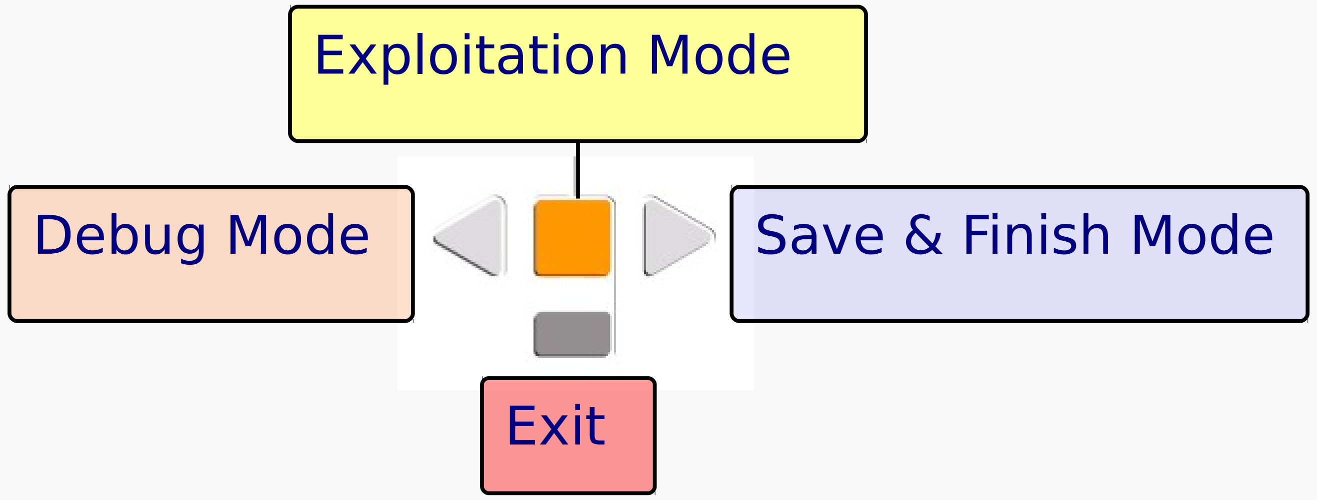
```
frequentist_data(:,1,:) = [
66 2 3 0
0 10 0 0
2 0 5 0
0 0 0 6
]; % [s,a,sp]
frequentist_data(:,2,:) = [
69 0 1 0
5 3 1 0
55 0 13 0
5 0 8 0
]; % [s,a,sp]
frequentist_data(:,3,:) = [
218 2 0 0
46 11 0 0
3 0 3 0
6 5 0 0
]; % [s,a,sp]
frequentist_data(:,4,:) = [
321 44 47 23
0 8 0 1
0 0 3 0
0 0 0 5
]; % [s,a,sp]
```

UNIVERSIDAD DE MÁLAGA
Departamento de Ingeniería de Sistemas y Automática

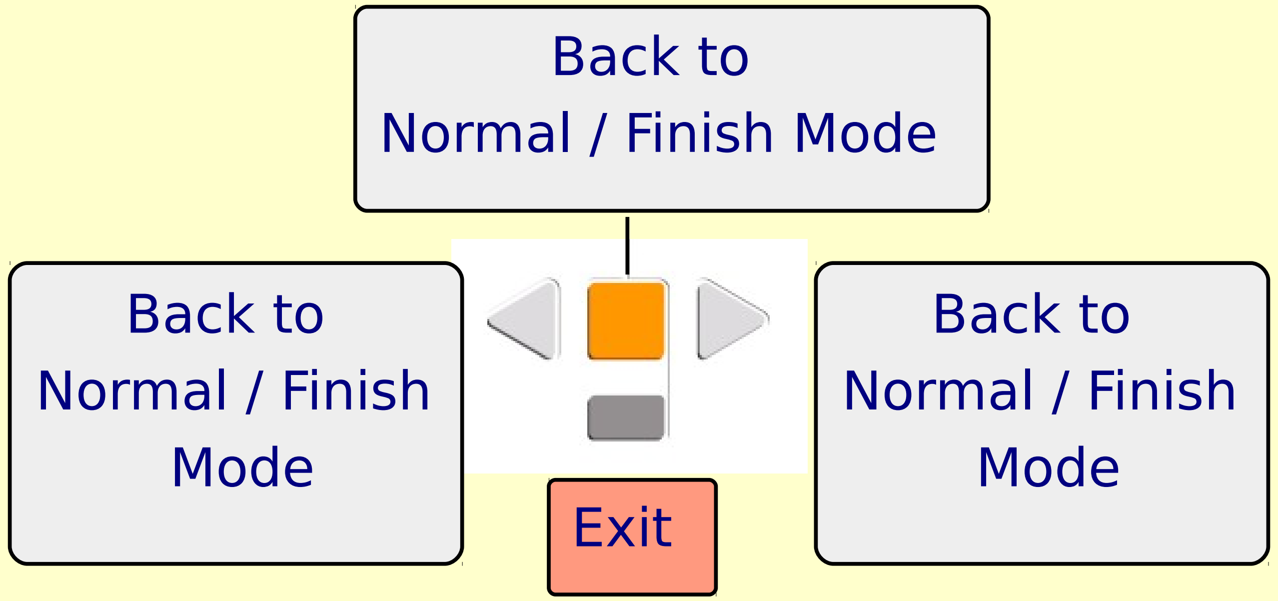
NXC Template: Operating modes

(Q_learning.nxc)

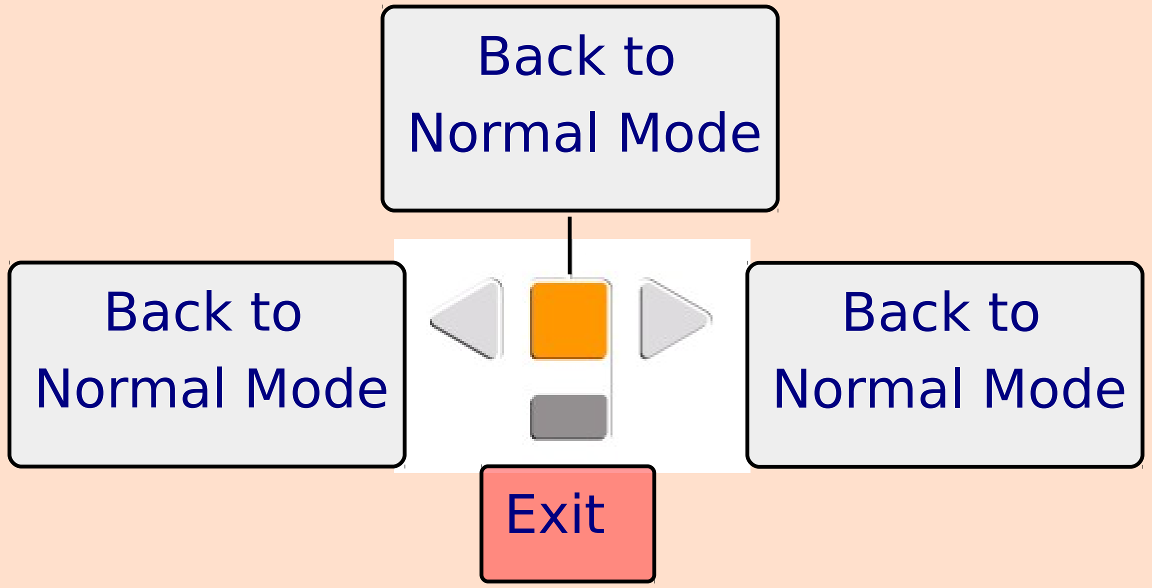
Normal Mode: Learning process (default)



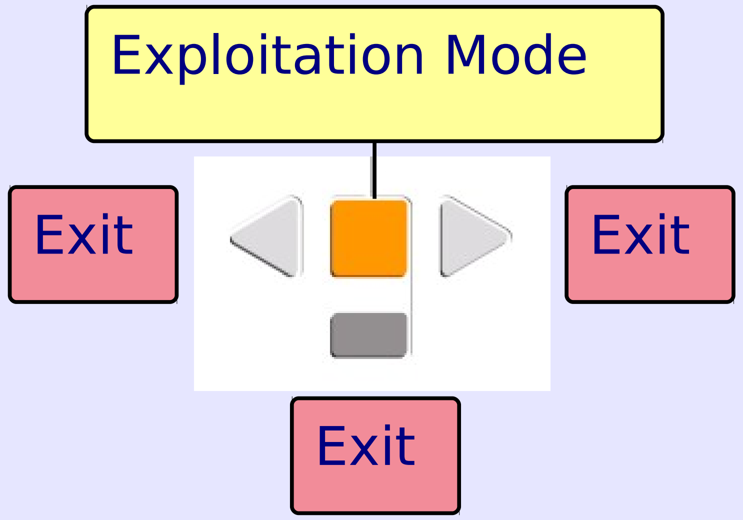
Exploitation Mode: Exploit policy (no learning)



Debug Mode: Learning process with long delay between steps



Finish Mode: Save and Wait



UNIVERSIDAD DE MÁLAGA
Departamento de Ingeniería de Sistemas y Automática

Example: Simple Wandering Task

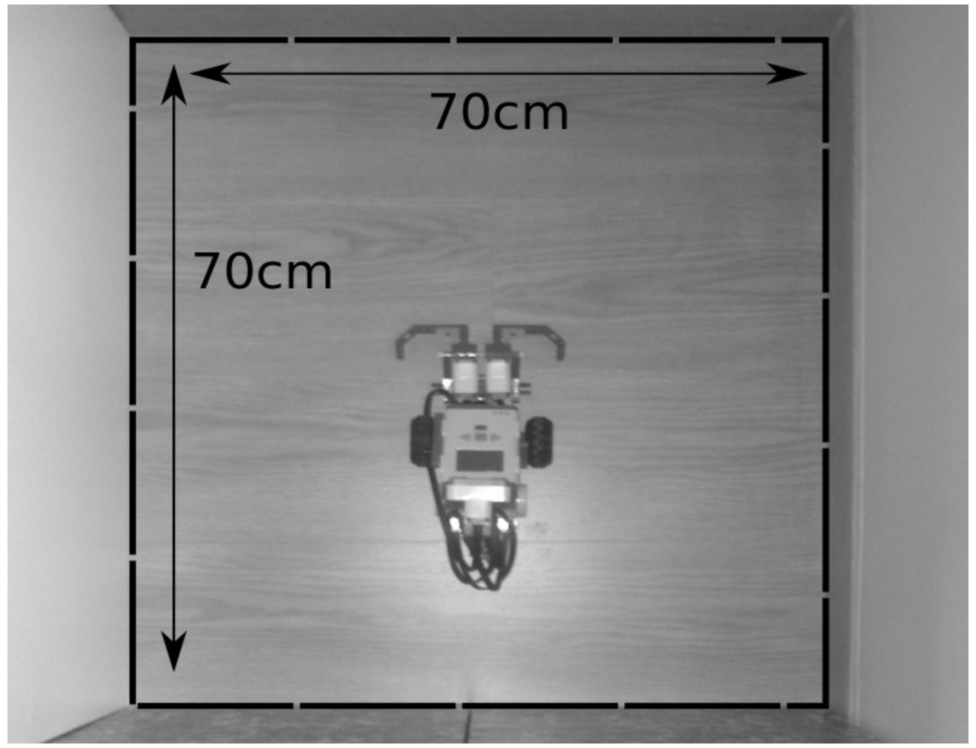
Goal: Wandering evading obstacles after touching them

States	
s1	no contact
s2	left bumper contact
s3	right bumper contact
s4	both contacts

Actions	
a1	stop
a2	left-turn
a3	right-turn
a4	move forward

Optimal	
s1	a4
s2	a3
s3	a2
s4	a2 or a3

Test Scenario




```
1: Read TASK.h from TEMPLATE folder.

2: Open Bricxcc* and connect the robot NXT**.

3: Open learning.nxc from the TEMPLATE folder
    and download it to NXT.

4: Place the robot into the test scenario and
   execute the downloaded file.

4: Wait until the end of the learning process (<5min).

5: Check the resulted log-file.
```

* Bricxcc; 33810 (NBC/NXC 1.2.1 r5)

** NXT Firmware: V1.31.rfw

UNIVERSIDAD DE MÁLAGA

Departamento de Ingeniería de Sistemas y Automática

Practical Activity: Wander-8s

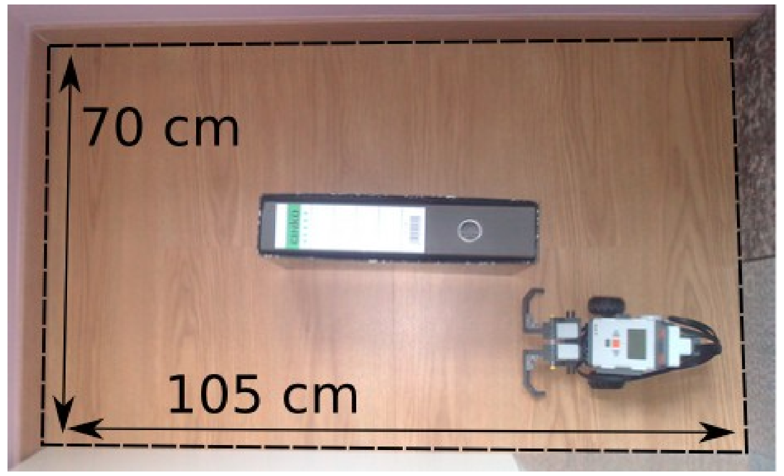
Goal: Wandering avoiding obstacles

States		
s1	no contact	- no obstacle
s2	left bumper contact	- no obstacle
s3	right bumper contact	- no obstacle
s4	both contacts	- no obstacle
s5	no contact	- obstacle
s6	left bumper contact	- obstacle
s7	right bumper contact	- obstacle
s8	both contacts	- obstacle

Actions	
a1	Stop
a2	left-turn
a3	right-turn
a4	move forward

Optimal	
s1	a4
s2	a3
s3	a2
s4	a3 or a2
s5	a3 or a2
s6	a3
s7	a2
s8	a3 or a2

Test Scenario



A1

```
1: Modify TASK.h from TEMPLATE folder:
   - observeState() must include ultrasonic sonar observations and a
     new discretization of states.
   - do not change GAMMA, INITIAL_ALPHA, or INITIAL_POLICY.

2: Execute the new program in the test scenario and check the results.

3: In case of anomaly behavior, modify TASK.h and repeat the previous
    step. MOTOR_POWER, THRESHOLD_DEGREES and THRESHOLD_DISTANCE could
    Be required to be tuned properly.
```

UNIVERSIDAD DE MÁLAGA
Departamento de Ingeniería de Sistemas y Automática

Practical Activity: Wander-8s

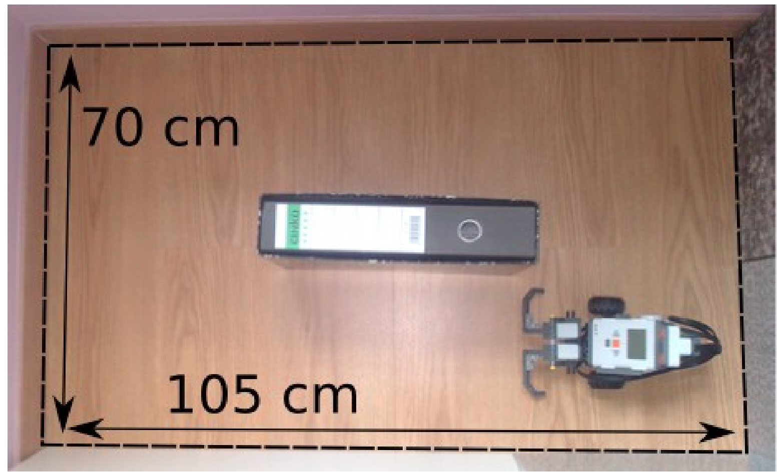
Goal: Wandering avoiding obstacles

States		
s1	no contact	- no obstacle
s2	left bumper contact	- no obstacle
s3	right bumper contact	- no obstacle
s4	both contacts	- no obstacle
s5	no contact	- obstacle
s6	left bumper contact	- obstacle
s7	right bumper contact	- obstacle
s8	both contacts	- obstacle

Actions	
a1	Stop
a2	left-turn
a3	right-turn
a4	move forward

Optimal	
s1	a4
s2	a3
s3	a2
s4	a3 or a2
s5	a3 or a2
s6	a3
s7	a2
s8	a3 or a2

Test Scenario



A2 Perform 3 experiments (<10 min per exp) and save their log-files.

A3 Check the results. In case of failure in the learning process, analyze the problem and implement a solution. At least one correct learning process should be achieved.

Propose and implement at least one improvement to the learning process implemented for this task (free-choice). Repeat A2 with the improved method. Compare the results.

Practical Activity: Wander-8s. Group Work

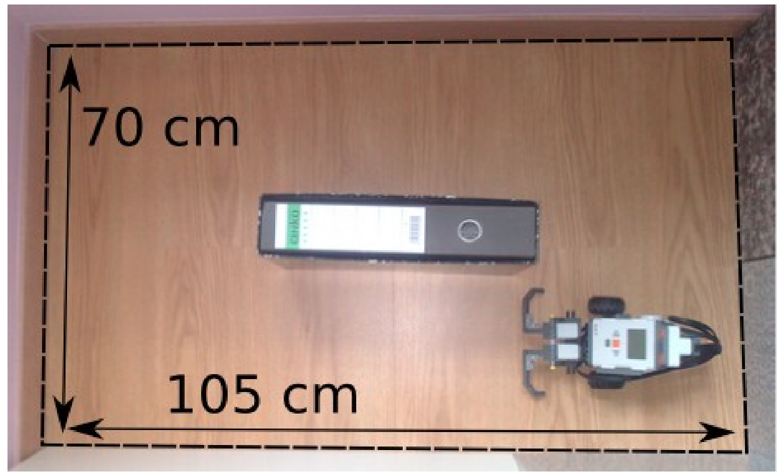
Goal: Wandering avoiding obstacles

States		
s1	no contact	- no obstacle
s2	left bumper contact	- no obstacle
s3	right bumper contact	- no obstacle
s4	both contacts	- no obstacle
s5	no contact	- obstacle
s6	left bumper contact	- obstacle
s7	right bumper contact	- obstacle
s8	both contacts	- obstacle

Actions	
a1	Stop
a2	left-turn
a3	right-turn
a4	move forward

Optimal	
s1	a4
s2	a3
s3	a2
s4	a3 or a2
s5	a3 or a2
s6	a3
s7	a2
s8	a3 or a2

Test Scenario



B1

Share and discuss the improvements and results. Select a robot and implement the best solution proposed (free choice). Repeat 3 experiments and discuss the results.

B2

Collect the NXC source code resulted in B1 (modifications in the template must be commented), the log-files obtained and write a brief report about the work performed and the results. Upload them to the wiki of Cognitive Robotics course. (http://mop.cv.uma.es)

UNIVERSIDAD
DE MÁLAGA

Reinforcement Learning in the Lego® Mindstorms® NXT Robot

Ángel Martínez-Tenor

Practical Activity – Cognitive Robotics

Master Degree in Mechatronics Engineering

2015-2016

University of Málaga. System Engineering & Automation

Departamento de Ingeniería de
Sistemas y Automática

V1.1 January 4, 2016

UNIVERSIDAD
DE MÁLAGA