**Football Control with Roomba Robot through Deep Reinforcement Learning**

**DU Yansong, HUANG Yeshun, XUE Junyuan, Project 2 Proposal**

**Why:**

The first and main reason behind the selection of this project topic is that we all love playing football. So why not let the robot have a try? Since football is a truly complicated tasks, which includes collaboration, attacking and defensing, it is hard to complete all the tasks involved in the real football match in this project by the traditional method. One of the first tasks that every football player learns are to move the ball to some position, so we decide to train the robot to accomplish the same task: carrying the football into the destination position. This is where our project comes.

**Conventional Algorithms:**

 Traditionally, for controlling the football, one way to design the program is to simply kick the football directly towards the goal. This design would be problematic in the cases of either the target state becomes unreachable due to the change of the environment, or the parameters of system changes such as the mass of the football, or the friction of the environment. For instance, when the football locates at the corner of the room, the robot is commanded to move to the position at the back of the football towards the target point, which is not accessible anymore. Also, when a new robot or football is used, the program has to be changed again. In short, the conventional methods are lack of flexibility and adaptability.

 In conventional algorithms, these problems can only be solved in the cost of adding more commands under discrete situations, which makes the whole program rather cumbersome. On the contrary of this, when the deep reinforcement learning is utilized for the robot to learn how to kick or dribble the football to the target position, the robot should be able to act differently based on different system, and efficiently according to the definition of the reward and cost function.

**Problem Statement:**

 A robot is playing with football and it wants the football to enter into the gate. There are two ways for it to complete the goal- shooting or dribbling. Undoubtedly, it is most time-efficient for the robot to shoot, but there is some sliding when robot tries to kick the football. In this way, the football will not move as expected. So, what is the robot’s strategy now?

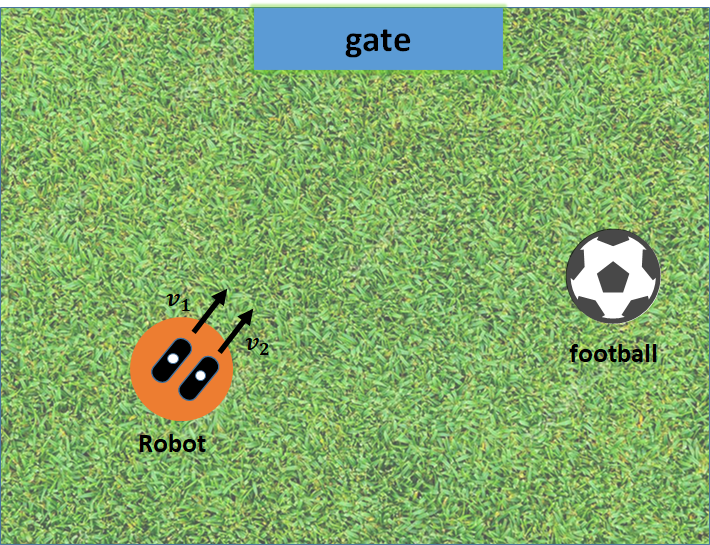


Figure 1: Roomba robot tries to kick the football into the gate

 It is a world based on velocity instead of acceleration. The action of robot is the velocity of two wheels, which will change the velocity and orientation of the robot. When the robot hits the football, the new velocity of the robot and the ball will be calculated by law of conservation of momentum (1) and law of conservation of mechanical energy (2).

 Simulation will be set up in pyglet which is a rather simple and powerful Python library in game development. Further, this UI will not only provide a better presentation, but it will help us to debug the problem directly as well.

**RL Cast:**

 State Space: The state space includes:

two-dimensional positions (x, y) of the football,

two-dimensional velocities (vx, vy) of the football,

three-dimensional positions (x, y, θ) of the robot,

three-dimensional velocities (vx, vy, ω) of the robot,

two-dimensional position (x, y) of the gate.

 Action Space: The action space includes:

the velocity commands (accelerate or decelerate or stay) of the two wheels of the robot.

 Reward Structure:

+100 when football reaches the target

-0.2 for every time step

(Optional) -10 for robot collides with the wall

(Optional) +0.1 for distance between the football and the gate shorter

(Optional) +0.1 for distance between the football and the gate longer

 Neural Network Structure: Currently fully connected network is planning to be used. The exact structure of this neural network is however highly flexible and to be changed in terms of the number and sizes of layers according to the actual situation.

**RL algorithms and envisioned results:**

 In this project, different RL algorithms will be attempted. Firstly, A2C (advantage actor-critic) will be utilized since it is an efficient RL learning method for continuous control, and it is relatively well developed. Also, DDPG (Deep Deterministic Policy Gradient) will be attempted. DDPG combines actor-critic approach with the DQN algorithm, so that it is able to deal with continuous action space. Last but not least, PPO will be tried also.

 We will compare the performance of the training results with A2C method, DDPG method, and the traditional method, especially in the aspect of the task accomplishment rate (how many times the agent succeeds or fails in learning the task), time consumption during training process, and task performance.

 The training will be tested in the environment as described above. The testing results of all the methods will be discussed in details.

Details:



wheel velocity (v1, v2) 🡪 robot velocity (vx, vy, ω), using orientation θ

orientation θ, wheel velocity (v1, v2) 🡨 robot velocity (vx, vy, ω)

1 When the robot hits the football

For 2 dimension vectors, the cross product is:

through rotation matrix

Conservation equation

through rotation matrix

2 When the football or the robot hits the wall

If hitting the wall in x axis:

If hitting the wall in y axis:

Test enviroment,

when mass rate=1

[(-1.41835696349365e-14, 26.3286791132020), (26.3286791132020, -1.41835696349365e-14)]

Due to 2 orders, need to select the solution. Parently, one solution is equal to the original value, so this solution needs to be recognized and abandoned.

[(90.0244021966097, 181.867479185068), (91.8430769884604, 2.31777663704753e-12)]

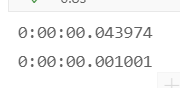
1 [91.84307698846038, 39.55817499950837] , [0.0, 0.0]

2 [90.0244021966097, 39.55817499950837] , [181.867479185068, 0.0]

The 0.001 is a small error estimated, since there are some computation error. The value works well, from the data.

Limit value, unless the speed will be very great. [-17228.9961713056, 37017.49304034878] , [-45832.0427636374, 16309.2816964296]

The solve speed is too slow, so directly give the solution equation. Not using Newton way to solve.



[(-3.83419711291238e-13, 245.940627825647), (245.940627825647, -3.83419711291238e-13)] 0:00:01.486881

0.0 245.94062782564743 245.94062782564743 0.0 0:00:00

1 [245.94062782564743, -72.83857209423137] , [0.0, 0.0] , 1.4650000000000063

2 [-3.83419711291238e-13, -72.83857209423137] , [245.940627825647, 0.0]

[(-148.014967762193, 206.233879262399), (206.233879262399, -148.014967762193)] 0:00:00.332043

-148.014967762193 206.233879262400 206.233879262400 -148.014967762193 0:00:00.002038

1 [206.23387926239954, -76.1353060334861] , [-148.014967762193, 196.413751386278] , 1.265000000000006

2 [-148.014967762193, -76.1353060334861] , [206.233879262399, 196.413751386278]

[(-117.119731342851, 10.6556409509985), (10.6556409509985, -117.119731342851)] 0:00:00.313175

-117.119731342851 10.6556409509986 10.6556409509986 -117.119731342851 0:00:00.002040

1 [10.655640950998631, -226.1973943893073] , [-117.119731342851, -259.603049326133] , 1.1050000000000046

2 [-117.119731342851, -226.1973943893073] , [10.6556409509985, -259.603049326133]

[(-258.684860207078, 59.7235037371086), (59.7235037371086, -258.684860207078)] 0:00:00.299043

-258.684860207078 59.7235037371087 59.7235037371087 -258.684860207078 0:00:00.002002

1 [59.72350373710866, -259.43439718979556] , [-258.684860207078, -24.2781589736144] , 2.6300000000000043

2 [-258.684860207078, -259.43439718979556] , [59.7235037371086, -24.2781589736144]

[(-62.3586171224823, 53.4662327943304), (53.4662327943304, -62.3586171224823)] 0:00:00.329292

-62.3586171224823 53.4662327943303 53.4662327943304 -62.3586171224823 0:00:00.002051

1 [53.46623279433035, -381.1341209790514] , [-62.3586171224823, 16.3624195141236] , 3.3550000000000044

2 [-62.3586171224823, -381.1341209790514] , [53.4662327943304, 16.3624195141236]

[(-45.7306465069719, 405.537811828624), (405.537811828624, -45.7306465069719)] 0:00:00.268960

-45.7306465069713 405.537811828623 405.537811828623 -45.7306465069713 0:00:00.002039

1 [405.53781182862343, -283.5546202637172] , [-45.7306465069713, 32.1725782558399] , 3.5550000000000046

2 [-45.7306465069719, -283.5546202637172] , [405.537811828624, 32.1725782558399]

[(-344.297294522669, 203.234231597039), (203.234231597039, -344.297294522669)] 0:00:00.296000

-344.297294522668 203.234231597038 203.234231597038 -344.297294522668 0:00:00.001051

1 [203.23423159703808, 228.3849806777852] , [-344.297294522668, 216.691865557403] , 4.010000000000004

2 [-344.297294522669, 228.3849806777852] , [203.234231597039, 216.691865557403]

[(-207.917395541933, 331.771707647700), (331.771707647700, -207.917395541933)] 0:00:00.306040

-207.917395541933 331.771707647700 331.771707647700 -207.917395541933 0:00:00.001996

1 [331.77170764770045, 268.26368063054247] , [-207.917395541933, 212.202436655768] , 5.675

2 [-207.917395541933, 268.26368063054247] , [331.771707647700, 212.202436655768]

[(-393.784014508514, 210.670237269023), (210.670237269023, -393.784014508514)] 0:00:00.293995

-393.784014508514 210.670237269023 210.670237269023 -393.784014508514 0:00:00.002059

1 [210.67023726902295, 355.1164998344146] , [-393.784014508514, -6.04069827711972] , 5.81

2 [-393.784014508514, 355.1164998344146] , [210.670237269023, -6.04069827711972]