Robot Position control in pipes using Q Learning

Danilo Sulino Silveira Pinto*, Karina Rocha Gomes da Silva
Escola de Engenharia Elétrica, Mecânica e de Computação
Universidade Federal de Goiás - UFG
Goiânia – Goiás - Brazil
*sulinod@gmail.com

***Abstract*— In the most critical hydro crisis in Brazil, 37 percent of the whole amount of treated water is wasted before reaching consumers. A robot with a position control to travel inside a pipe is an important step in the pursuit of an autonomous solution to detect and correct pipes failures. This paper shows a Q Learning controller algorithm implemented using a microcontroller in a mechanical body of a commercial pipe inspection robot. Using only the measurements of a gyroscope, and controlling the wheels' motors on the left and right sides, the controller learned the best set of movements to ride inside a 300mm sewer pipe, in the tested conditions. Real tests in a 300mm pipe were performed using the developed algorithm and it was compared to a random movement and to a straight forward movement.**

Keywords—Pipe inspection; Position control; Q-Learning; Robot;

I. INTRODUCTION

According to a report from *Folha de São Paulo Journal* [1], in Brazil, during one of the most critical hydro crisis, near 37% of the whole amount of treated water for human consumption is wasted before it reaches the consumer's faucet, and the pipes failures of the water supply network are the main cause of this waste. The report also shows an estimative that, in 2010, the cost of this wasted water reached the impressive number of R$1.3 billion, or, at the time, the equivalent to US$650 million.

There are more than 2.6 million meters of sewage disposal system only at the Goiânia metropolitan region, in the State of Goiás, Brazil. And without instruments to verify the pipes' situation, their quality and conservation are unknown, leading to leaks and irregularities in large quantities, as it happens in the treated water pipes, according to the state sanitation company, SANEAGO [2]. And according to Romanova et. all [3], in 2012, the UK drainage system has over 300 000 km of pipes.

One way to detect this pipes failures is using robots, equipped with cameras that transmit the images in real time to the operator who is the responsible to detect the cracks. However, the long extension pipes make this process expensive and very slow. So it would be of great value if a robot could run along the sewage disposal system and the water supply network autonomously detecting by itself all the pipes problems.

Sponsors: RYD Engenharia and FAPEG

The position control of a robot inside a pipe is an important step in the pursuit of an autonomous solution to the water pipes waste. This paper used Q-Learning as the control method for the robot, which is a control learning method used in many areas including robotics. And the focus, observed in [4], on Reinforcement Learning as an approach to achieve optimal control encouraged the use of this method.

Fjerdingem et. all [5] analyzed the application of several reinforcement learning techniques for continuous state and action spaces to pipeline following for an autonomous underwater vehicle (AUV), and its results were validated on a realistic simulator of the AUV, and confirmed the applicability of reinforcement learning to optimize pipeline following behavior. Pipe tests performed by a robot using a Reinforcement Learning method can contribute to the science and social community.

The text has three more sections before the conclusions. The Q-Learning method is explained in section two. Section three shows the development of the algorithm and of the robot with all the details. And in section four the tests and results are detailed.

II. REINFORCEMENT LEARNING

Reinforcement Learning is a learn what to do method in which the learner must discover which actions maximize a numerical reward signal just by trying them.

As told in [6], this method is distinguished from other computational approaches due to its emphasis on learning from its direct interaction with the environment, without relying on exemplary supervision or complete models of the environment.

An agent interacts with a stochastic process modelled as a Markov decision process (MDP), and can observe the current state and immediate reward [5]. So, in a real application the agent must be able to sense the state of the environment, according to its needs, and must be able to take actions that affect the state. Also, the agent must have a goal or goals relating to the state of the environment. The method formulation is intended to include just these three aspects: the sensation of the environment, the action that it takes and the goal.

Supervised learning is the kind of learning studied in most current research in machine learning, statistical pattern recognition, and artificial neural networks. However, it is

different from Reinforcement Learning because the agent learns from examples provided by a knowledgeable external supervisor. In spite of the fact that it is an important kind of learning, it is not adequate for learning from interaction, because in interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act.

One challenge in the method is the correct balance between exploration and exploitation. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future.

A. Q-Learning

One of the most important breakthroughs in reinforcement learning was the development of an off-policy TD control algorithm known as Q-learning (Watkins, 1989). It can be used to find an optimal policy of actions to any Markov Decision Process (MDP).

This method brings the advantage of being independent of an environment model, as it updates its estimative based on its own estimative. Because of that, this method is in a subdivision inside Reinforcement Learning called Temporal-Difference Learning.

This method needs: A matrix S, that represents the possible states of the robot; An action matrix A(s), that contains all possible actions that the agent could take in each moment; A reward matrix R(s,a,s') that would guide the robot to the goal as it pursues to maximize the reward; And a matrix Q(s,a), that is an Action-Value function, that indicates which action the robot should take in each state it arrives. These matrix are dependent on the current state, s, and the action to be taken, a.

Only in a simulation case a state change model T(s,a,s'), which would be a series of equations that would simulate the environment, is necessary, to perform the actions.

The updates and equations about these matrix can be written more specifically using the next state as, s' or $s_{\{t+1\}}$, the next action to be taken as, a' or $a_{\{t+1\}}$, and the upcoming reward as, r' or $r_{\{t+1\}}$.

In the method S, A and R are defined and Q is initialized arbitrarily, and according to the movements the robot is performing, the agent receives rewards and tries to maximize them, thus learning the best movements to be taken in each state to reach the goal.

Iteration methods have difficulties regarding the time and high number of actions that the agent need to realize to learn the optimal set of actions that should be taken in each state, which is called policy.

The optimal policy can be reached updating the matrix Q according to the equation 1.

$$Q(s,a) = Q(s,a) + \alpha[r' + \gamma \max Q(s',a') - Q(s,a)] \quad (1)$$

The variable α is the learning rate ($0 < \alpha < 1$) that is used to decide how much previous learning is retained. The variable γ is the discount factor, which commands the preference of immediate reward, r', (smaller value γ) or future reward (larger value of γ) [4].

The algorithm to implement a Q-Learning method is shown in Figure 1.

```
1 Initialize Q arbitrarily
2 Repeat (for each episode):
3     Initialize S
4     Repeat (for each step in the episode):
5         Choose a from s using policy derived from Q
6         Take action a, observe r, s`
7         Q(s,a)←Q(s,a)+α[r + γ max_a, Q(s',a') - Q(s,a)]
8         s←s';
  until s is terminal
```

Fig. 1. Q-Learning Algorithm

III. DEVELOPMENT

Trying to develop a robot and an algorithm as easiest as possible to be used in a real application, an Arduino UNO [7] was used as the microcontroller, and a MPU6050 [8] as sensor, which are cheap and easy to find components. And with the same focus, the tests were performed by freely hands, without using precise positioning instrumentation.

A. Algorithm

The objective of the algorithm is to make the robot learn to stay at the perfect horizontal position riding forward using only the angles measured in one Cartesian plane direction. The plane direction of the measured angles is shown in Figure 2, to a better understanding.

The code was developed based on the Q-Learning algorithm shown in Figure 1, in an Arduino microcontroller specific language, which is similar to the C language. The Arduino UNO was the microcontroller platform used, and the MPU6050 the sensor.

The matrix Q was initialized arbitrarily with the same value in all the positions, 9. The action matrix (A) was created based on the velocities that could be applied in the left and right side motors, using a Pulse Width Modulation (PWM), according to table I. The state matrix (S) was developed based on the gyroscope measures with the values divided in 13 states as shown in table II.

The goal of the robot is to arrive at the horizontal, which means arrive at the state 1. Also, the states 6, 7, 12 and 13 should be avoided as they can lead the robot to roll over.

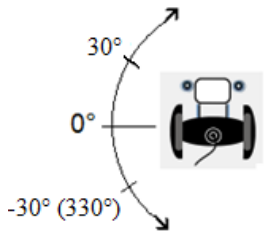

Fig. 2. Angles measured based on the back view

The PWM modulation allows the controller to control the voltage delivered to the load by varying the pulse width inside a cycle period. For instance, to have half voltage, the pulse width should be reduced to half period.

Initially the PWM values were 0, 125 and 255, from the maximum of 255, which would result in a voltage delivered to the load of 0%, 49% and 100%, respectively. However, the motor could barely move with a 49% voltage due to its low torque. The high reduction of the motor made the 0% velocity lock the wheels, and stopped the robot even when the other side tried to move. So, the PWM values to control the motors used were 90, 175 and 255 as minimum, medium and maximum speed, which would deliver a voltage of approximate 35%, 69% and 100%, respectively.

TABLE I. ACTIONS WITH LEFT AND RIGHT MOTOR'S SPEED.

Action	Left	Right
1	Maximum	Maximum
2	Minimum	Medium
3	Minimum	Maximum
4	Medium	Minimum
5	Medium	Medium
6	Medium	Maximum
7	Maximum	Minimum
8	Maximum	Medium

The first test was performed in an infinite time episode, in which the robot followed the pipe indefinitely. However, this endless episode led the robot to a vicious movement, as it achieved the goal and did not have the opportunity to experience other relatively distant states, so these first test results were not considered in this work.

TABLE II. STATES SECTIONS

State	Minimum	Maximum	Range
1	-1°	1°	2°
2	1°	5°	4°
3	5°	9°	4°
4	9°	16°	7°
5	16°	26°	10°
6	26°	56°	30°
7	56°	90°	34°
8	355°	359°	4°
9	351°	355°	4°
10	344°	351°	7°
11	334°	344°	10°
12	304°	334°	30°
13	270°	304°	34°

In the second set of tests, episodes of 30 actions each were implemented. In this way the robot was positioned in a random position, so that it could try, in 30 actions, to reach the objective, thus updating the Q matrix.

A good action that leads to the goal receives a reward, which leads it to a higher possibility to be chosen. So, as in most of the learning methods, a random movement was implemented, in the way that in 10% of the time the actions taken would be random and not chosen, trying to avoid the choice of a good action that leads to the goal just because it was tried first instead of the optimal action. This means 10% of exploration and 90% of exploitation.

The tests were performed in a half pipe of a water collection pipe, therefore without having the microcontroller attached to a computer. In this way, the information of the matrix Q and the number of episodes tested were stored in Arduino's EEPROM.

The sensor used communicates with the microcontroller using I2C (Inter-Integrated Circuit), a serial multi-master bus developed by Philips [9]. Due to magnetic interference from the environment and lack of precision of the sensor, letting the sensor static in a surface, the measurements can vary 2 degrees randomly for more or less. So, a buffer was implemented to capture 100 measurements and use the mean as the real value received by the sensor, and 0.09 of maximum variation was achieved, in the same static testing conditions.

B. Robot

The robot used was the model VX1-300, showed in Figure 3, a commercially available inspection robot which was a courtesy from RYD Engenharia [10]. It is a robot to run in pipes that have more than 300mm of diameter. And there are four continuous current 24v motors – one for each wheel –, which have maximum torque of 5.5Kgf.cm and 29 rpm (Rotations per Minute).

The microcontroller, using two H-bridges, controls the two motors of the right side together, as well as in the left side. The sensor was attached to the top surface. And two 12v motorcycle batteries were used to supply the system, together with a voltage regulator to supply the Arduino with 9v.

Fig. 3. Robot VX1-300

IV. TESTS AND RESULTS

A. Learning test

The tests were performed in a water collection half pipe of 300mm of diameter, as shown in Figure 4. The characteristics of this pipe fit to a real situation, made of concrete as most of this size pipes and dirty as any pipe that needs inspection.

Along with the fact that it was not a closed pipe, it was easy to supervise the tests.

Accordingly to the angles reference cited before, if the robot is inclined 10°, it can be noticed that if it turns the left side motors faster, it will turn to the right tending to distance from the more inclined surface of the left side. However, as the tube constrains the robot in both sides, it is necessary to verify the development of the algorithm before assuming the best set of actions.

Fig. 4. Half pipe used in the tests

The table III shows the Q matrix of the 30 actions per episode test. It performed 690 actions or 23 episodes. For a better understanding the motor commands for each action were shown, and the back colors of the State number is brighter as it is closer to the goal state. The positions in the Q matrix represent the values of each action (column) in each state (row) that the robot can be in. The values with yellow background color are the highest action value of each state.

It was possible to see an addiction in the robot's first test, as there were few positions with high values. The best action for the robot at the goal position is to ride forward, so it received a value more than the double of the value of all the other positions. Also the values did not have an easy pattern to be recognized in the movements, for instance, in one case two states with similar angles, had opposite actions as the best option.

TABLE III. Q MATRIX (STATE \ ACTION)

A / S	1	2	3	4	5	6	7	8
1	12.18	8.90	9.00	9.00	9.00	9.65	9.00	9.00
2	8.89	8.67	8.75	10.02	8.80	8.89	8.72	12.65
3	8.89	8.72	8.89	8.89	9.00	8.51	8.81	10.02
4	8.67	8.75	8.51	8.80	8.89	8.72	9.65	9.00
5	8.67	8.75	10.02	8.80	8.89	8.72	12.65	9.00
6	8.72	8.89	8.89	9.00	8.51	8.81	10.02	9.00
7	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
8	8.10	8.07	8.81	8.10	8.26	8.10	8.51	7.94
9	8.10	8.89	8.10	8.81	8.81	8.02	8.23	7.98
10	7.28	8.09	8.10	8.10	8.10	8.89	8.69	8.14
11	8.90	9.00	9.00	9.00	9.65	9.00	9.00	9.00
12	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
13	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
Left	Max	Min	Min	Med	Med	Med	Max	max
Right	Max	Med	Max	Min	Med	Max	Min	med

In table III it can be noticed that the robot tried great part of all the possibilities of actions in each state, and that the values are not very different from each other as in the first test. With a greater number of actions the values of each position can be better evaluated, which would consequently demonstrate a bigger differentiation in the values, with better precision.

The data observed in table III has a visible pattern. If the angles are positive, the best action is to move the left side faster than the right side, as can be seen in states 2, 3 and 4. In addition, if the angles are negative, the best action is to move the right side faster, as can be seen in states 6 and 7.

It is also possible to notice that in this interactive method, the number of actions needed is large and the time spent to learn the best set of movements is long.

With the test results, it is important to compare the learned method with other ones to make sure that the learning algorithm really works. So, comparative tests were performed comparing the Q-Learning method to a random movement and to a straight forward movement. After each action the robot should stop and verify the state that it reached.

A set of fifty actions was established, and three tests were made with each method, one with the robot with the right side higher, other with the left side higher, and other in the horizontal. And the actions were chosen by each method. As the tests were performed positioning the robot with free hands, the horizontal tests initialized in state 2, as it is difficult to position it in the exact horizontal.

The tables IV and V show the results of these tests. Table IV shows the number of actions in which the robot remained at the goal state. As expected, the Q-Learning method was superior to the other methods, so, in the three cases, the robot remained longer time at the goal position. The table V shows the action number in which the robot reached the goal for the first time. Once again, it is possible to see that the Q-Learning method tended to be better, although the results were not too different and not always better than the other methods as it happened in the table IV.

TABLE IV. NUMBER OF ACTIONS AT THE GOAL

Tests	Horizontal	Left	Right
Q-Learning	30	19	11
Random	13	0	4
Forward	18	0	1

TABLE V. REACHED THE GOAL FOR THE FIRST TIME

Tests	Horizontal	Left	Right
Q-Learning	3	20	17
Random	3	28	46
Forward	2	27	30

The tests state sequences are shown in Figures 5, 6 and 7. The first one shows the sequences which began near the horizontal. The second one shows the sequence which started with the left side higher. And the Figure 7, the third test. These figures show that the Q-Learning method assures that the robot

will reach the goal position no matter the initial state is, differently to the non-intelligent methods.

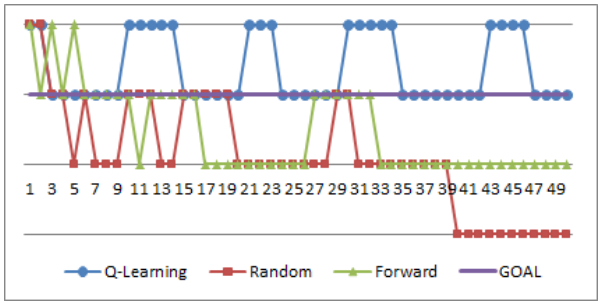

Fig. 5. Sequence of actions starting near the horizontal

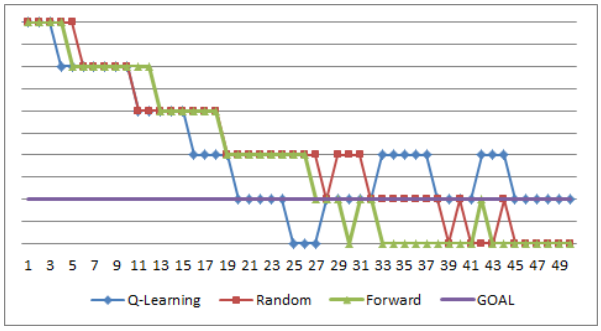

Fig. 6. Sequence of actions starting with left side higher

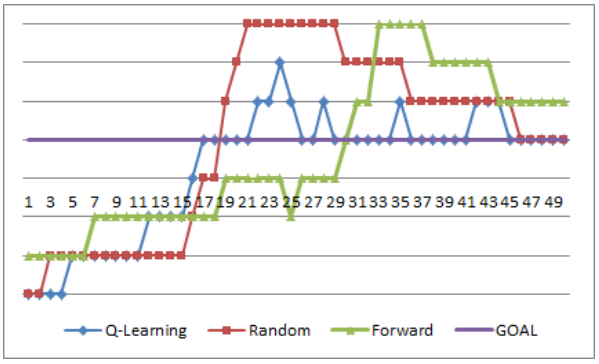

Fig. 7. Sequence of actions starting with right side higher

V. Conclusion

This paper analyzed the application of a Q-Learning algorithm in a pipe inspection robot. Real experiments have shown Q-Learning to be a good method to apply in a real world application as the example of robotics. It also showed that is relatively easy to implement this learning method as it does not need a mathematic environment model.

As the method needs a matrix with its size dependent to the number of states and actions, the memory required to process the algorithm and the number of actions needed for the agent to learn can increase almost exponentially due to the large number of actions and states needed in the situation.

In accordance with the results of [5], the real world experiments showed that reinforcement Learning is well suited to optimize pipeline following behavior for an inspection robot, and showed to be better when compared to a non-intelligent method.

Acknowledgment

Our thanks to FAPEG for the research support. Also, our thanks to RYD Engenharia, that lent the robot as courtesy.

References

[1] F. Lobel, "Folha de São Paulo," 21 Janeiro 2015. [Online]. Available: http://www1.folha.uol.com.br/cotidiano/2015/01/1578007-brasil-desperdica-37-da-agua-tratada-aponta-relatorio-do-governo-federal.shtml. [Accessed 31 Março 2015].

[2] SANEAGO, "SANEAGO," Agosto 2004. [Online]. Available: http://www.saneago.com.br/site/?id=esgoto6&tit=esgoto. [Accessed 31 Março 2015].

[3] A. Romanova, K. V. Horoshenkov, S. J. Tait and T. Ertl, "Sewer inspection and comparison of acoustic and CCTV methods," Water Management, vol. 166, no. WM2, pp. 70-80, 2012.

[4] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," Annual Reviews in control, no. 36, pp. 42-59, 2012.

[5] S. A. Fjerdingen, E. Kyrkjebo and A. A. Transeth, "AUV Pipeline Following using Reinforcement Learning," in ROBOTIK 2010, Munich, 2010.

[6] R. S. Sutton and G. A. Barto, Reinforcement Learning: An Introduction, Cambridge, Massachusets: The MIT Press, 2005.

[7] Arduino, "Arduino UNO," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed 2 February 2016].

[8] Arduino, "MPU-6050 Accelerometer + Gyro," Arduino, [Online]. Available: http://playground.arduino.cc/Main/MPU-6050. [Accessed 2 February 2016].

[9] NXP Semiconductors, "I2C-bus specification and user manual," 2014.

[10] RYD Engenharia, "Produtos: Robôs de Inspeção," [Online]. Available: http://www.rydengenharia.com.br/web/robos-de-inspecao/. [Accessed 7 June 2016].

[11] L. Zheng and Y. Kleiner, "State of the art review of inspection technologies for condition assessment," Measurement, vol. 46, no. 1, pp. 1-5, 2013.