SVC-A2C - Actor Critic Algorithm to Improve Smart Vacuum Cleaner

1st Everton Lima Aleixo
Instituto de Computao
Universidade Federal do Amazonas
Manaus, Brasil
everton.aleixo@icomp.ufam.edu.br

2nd Juan Gabriel Colonna
Instituto de Computao
Universidade Federal do Amazonas
Manaus, Brasil
juancolonna@icomp.ufam.edu.br

3rd Raimundo da Silva Barreto
Instituto de Computao
Universidade Federal do Amazonas
Manaus, Brasil
rbarreto@icomp.ufam.edu.br

***Abstract*—This work present a new approach to develop a vacuum cleaner. This use actor-critic algorithm. We execute tests with three other algoritms to compare. Even that, we develop a new simulator based on Gym to execute the tests.**

***Index Terms*—reinforcement learning, actor-critic, vaccum cleaner**

I. Introduction

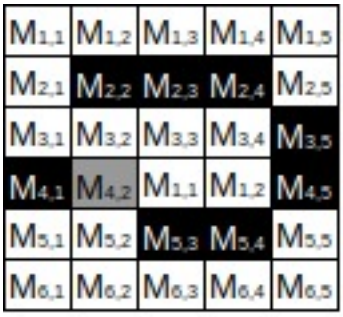
A vacuum cleaner is an embbeded system already part of daily life. These embbeded systens are getting smarter. The vacuum cleaner task can be defined as :

Given a region, a robot vacuum cleaner have to maximize the clean area avoiding the obstacles. Even that, it should minimize the time to do that.

The second objective is because as all IoT system, it suffer from energy capability problem. This problem have already be studied for a long time [1]–[4] and has a commercial appeal.

Most of researchers in this area, consider a region as a rectangular matrix, like illustrated in Figure 1 , where you have some cells that are empty (white), some that has obstacles (black) and one cell to represent the robot's start point (grey). The robot just can move to one of four neighboor cell per action. They consider as a good algorithm, the one that can pass by all empty cell using less steps.

Fig. 1. An area to vacuum cleaner represented as a matrix. White cell represent an space that the robot can pass, black ones are cells that has some obstacle, so the robot can not pass. And Grey cell represent the vacuum cleaner position.



This could be reductable to a travelling salesman problem (TSP). As TSP is a NP-Complete problem it just can solve LITTLE instances. So we should use some heuristic techinic to get some proximal better result.

In this assumption has another problem. It is intuitive that some parts of region tend to be more durty then others. And there are no certainty that the vacuum cleaner passing there will clean the space. This leads us to think that a good vaccum cleaner is not the one who travel for all region in less steps. We propose that a good one should clean the maximum of the region it can using less steps as possible.

The principal contributions of this paper are:

- A new approach to smart vaccum cleaner based on actor critic algorithm;
- a new way to verify the effectiveness of vaccum cleaner.

The rest of this paper is organized as follow: Section II give a little background about reinforcement learning and the actor critic algorithm. In Section III some works on smart vaccum cleaner are presented and discussed how they differ from this work. In Section IV the approach proposed is presented. The result are showed in Section V and finally in Section VI is presented a discussion about the new way to verify the effectiveness and fature works.

II. Background

In this section is presented a brief overview about reiforcement learning, in Section II-A, and the actor critic algorithm, in Section II-B. A more detailed description could be find in [5].

A. Reinforcement Learning

Reinforcement learning is a sub-area of machine learning where an agent interacts with the environment in exchange for a reward signal. The agent must learn how its actions affect the environment and try to maximise the reward [5]. With this, the agent learn what it should do without an explicit teacher.

Reinforcement learning algorithms estimate the value of a given state or the value of taking an action at a given state. After training, the agent will can decide the actions that it should take to maximise the total reward.

There are many methods for training the agent. In basic way, the agent create a table, known as *policy*, that has all possible states in lines and all possible actions in columns. So the agent in a state s executes an action a, receives a reward r (the value of the cell M_{sa}). The Figure 2 illustrates this

policy and a reward table. In deterministic environment the agent take the action that has major value of reward,

Fig. 2. **Left**: A policy example. **Right**: Reward of to be in a state.

	UP	DOWN	LEFT	RIGHT
$M_{1,1}$	-15	80	-50	30
$M_{1,2}$	-15	-20	60	40
$M_{1,3}$	-15	-18	55	45
...	...	...	...	...
$M_{2,2}$	-100	-100	-100	-100
...	...	...	...	...
$M_{6,6}$	10	-100	30	-100

Reward
1
1
1
...
-1
...
1

When the episode ends the values of table is updated within Equation 1, where V is the value of one cell in the table, R is the rewarding function and s' is the state the agent will be after execute action a.

$$V(s,a) \ = \ R(s) + \gamma \sum V(s') \tag{1}$$

Executing this procedure many times, the agent will learn what action give better reward in each state.

In the vacuum cleaner context there are more states that could be stored in a memory. For this reason it is necessary to use some approximate function to define the state. One approach that do this is the actor critic algorithm [6].

B. Actor Critic

The actor critic [6] algorithm is a reinforcement learning approach that mix value based mthod with policy based method. In its archtecture exist two agents:

- The **Actor** that controls how our agent behaves (policy-based);
- The **Critic** that measures how good the action taken is (value-based).

Generally, each agent is implemented as an approximation function. One function to give an approximation valeu of policy in a state s and the other to give an approximation value of a reward r that is gainned when the action a is taken in state s.

The most commom implementation of these approximation funtions is feedforward neural networks. To get good fuctions, these feedforward neural networks needs to be trainned. The trainning uses the following method:

1) Initially the actor read the environment, process it on its neural network, and choose the action that it think is the best;
2) Then, in the new state, the critic read the environment, process it on its neural network, and inform to the actor how good the action taken was.

Probabily the first actor actions will not be good, neither the feedback of the critic. But with many iterations, it tends to converge to good ones. It is like, a blind boy playing a video-game and he has a friend aside giving tips to him. So, when the boy take an action, he do not know if it was a good one, but his friend says if it gain points on the game or not.

This approach is really useful when it is not possible to map all states in a state's table. And this is the case of a vacuum cleaner robot, because depending on the size of the map, it will increasing the table size until it no longer fits in memory. As this work propose an algorithm that work with device with memory restrictions, make sense that this approach should be used.

III. Related works

The papers in this area are commumly divided in robots that can plan the path on known environment and on unknow one. The first ones are more restricted, because needs to be trained or programmed for each place.

Every related work use some representation of the environment. In general, the space is mapped as a matrix M (like Figure 1), where each cell M_{ij} is a discrete block representation of the space. Theses cells can be an obstacle or an empty space. All papers treat an empty space as a dirty space, so the best path planning is the one that pass by all empty space evicting the obstacles with less movements.

In this paper we propose that a good path planning is the one that can pass by the maximum cells that are really dirty and the robot can clean it, with minimum movements.

Leonard at al. [7] have used beacons sensors to define the path and [3] do a map of the environment based on SLAM. Many other papers [8]–[10] have used genetic algorithm to treat the problem. They implement the path as genes and try to get good solution for the problems of TSP. In this paper, the representation used was the image captured by a common camera RGB and are used to maximize the cleaning the durty area.

The usage of reinforcement learning to solve this problem already have been used. In [11] used monte carlo location and [12] used observable markov decision process. THE DIFFERENCE OF THIS PAPER IS THAT IT IS USING A MORE ROBUST REINFORCEMENT LEARNING, ACTOR CRITIC, THAT HAVE SHOW GREAT RESULTS IN GAMES AREA.

Other methods like bayesian network [1] and dynamic programming [13] have been already used too. In the next section the new approach is presented with details.

IV. SVC-A2C

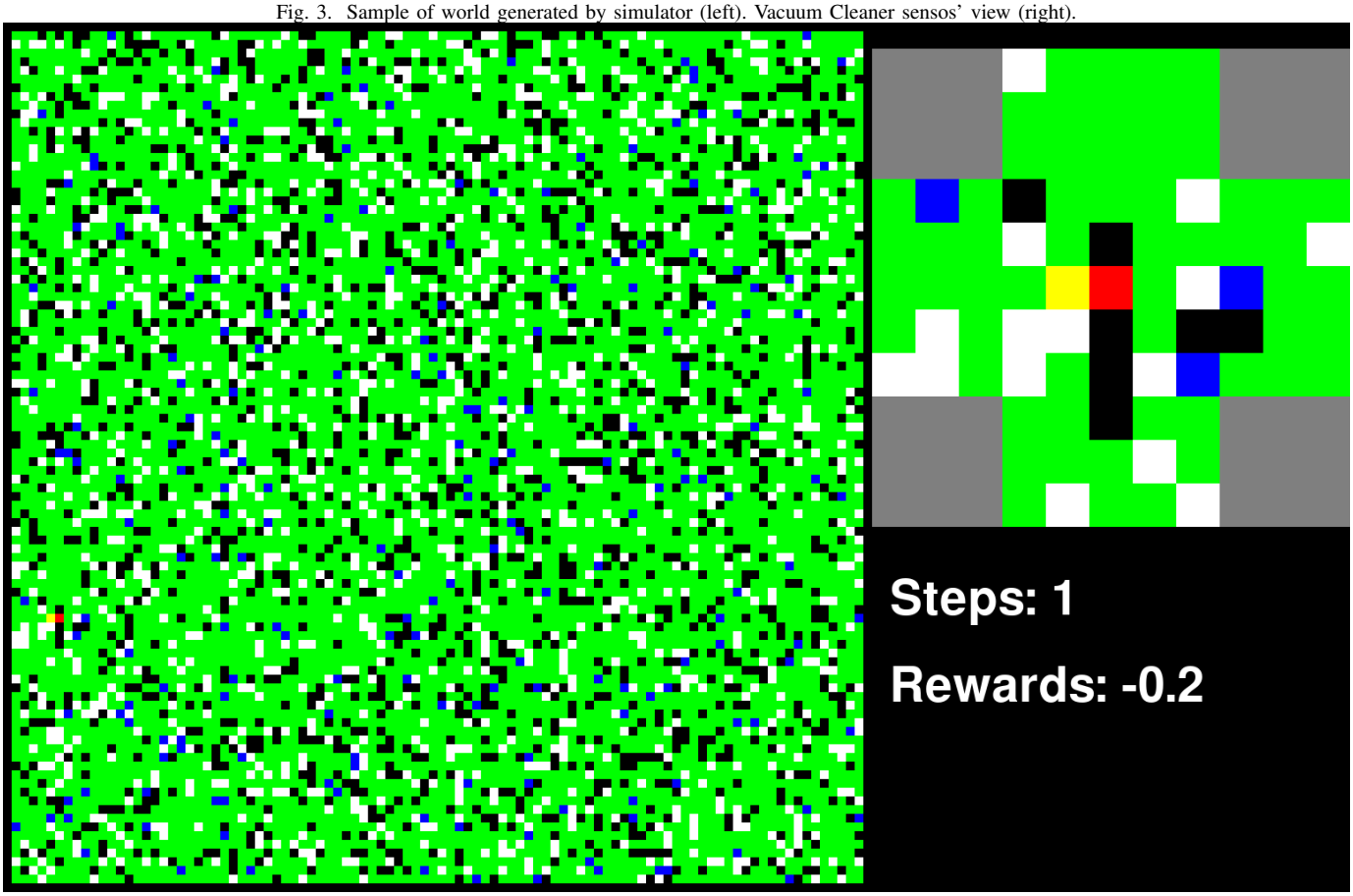
To develop the SVC-A2C, we first create a simulator. This was developed upon Gym [14]. The Gym is a Reinforcement Learning framework created by OpemAI to facilitate the improvements on this area. In Section IV-A is presented the details about the simulator and in Section IV-B is presented the agent used in this task.

A. Simulator

The simulator create a 2D world that is represented with a matrix, like a chess board. Each cell is a space of 20x20 centimeters. The vacuum cleanner occupy one cell.

The whole world is a map of 20x20 meters. So, it is a matrix of 100x100 cells. The visual sensor of the robot can

Fig. 3. Sample of world generated by simulator (left). Vacuum Cleaner sensos' view (right).



not observe all the space. It only can observe 5 cells in each direction and just can move one cell per time. The possible actions of the vacuum cleaner is:

- UP;
- LEFT;
- RIGHT; and
- DOWN.

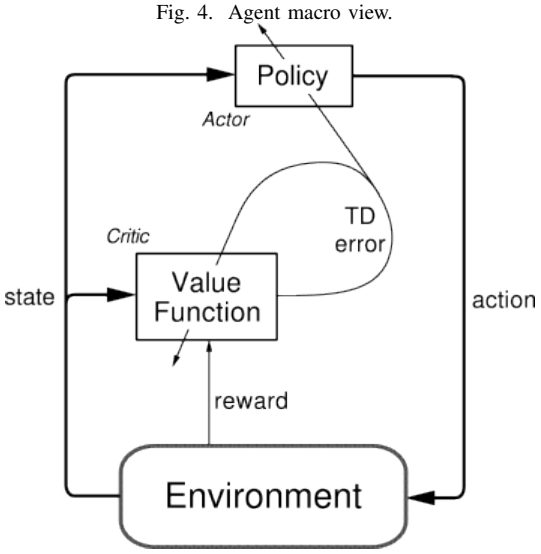
Even that, the robot can decide to auto turn off. It is penalized if there are a big area dirt yet. The cells have the following states:

- **Black:** It is a blocked space, like a wall ;
- **Blue:** It is a dirty area, but it is so dirty that the vacuum cleaner should avoid this area, because can not clean it;
- **Green:** This is the cells that has dirty that the vacuum cleaner can clean;
- **White:** This is clean areas, do not have job to be done there;
- **Yellow:** It is cells that the robot already has passed there;
- **Red:** It represent the position of the vacuum cleaner;
- DOWN.

In Figure 3 is presented a visualization of the world generated in simulator (on the left side of image) and the robot sensors' view (on the right side of image). The main objective is never go to a black cell, avoid to pass in white, yellow and blue cells and get the maximun green ones. In the next section the actor-critic agent is presented in detail.

B. Agent

Fig. 4. Agent macro view.



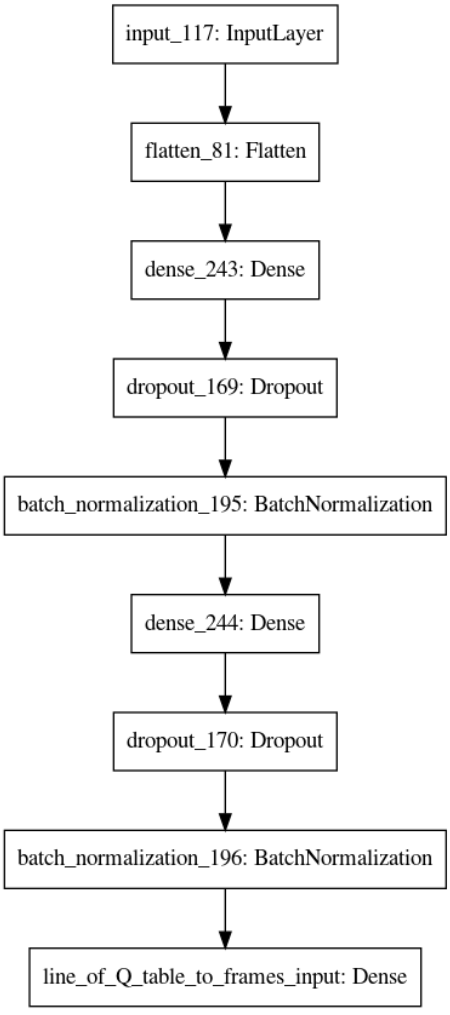
The agent is following actor-critic algorithm. The Figure 4 is presented a macro view of the algorithm. The agent has two component that interact with an enviroument. The **actor** is the component that learn a policy to decide what action the agent should take. After the action is taken, the enviroument generate a new state and a reward. This reward is used by the other component, the **critic**, that use this to learn the value of the state given that action.

Each state is represented by four stacked views (frames) of the robot. To do this, first we convert the image view of the sensors' robot to grey scale. Then take an action and apply the first step again. We do this four times. If the robot was startting the episode all the stacked frames is equal.

With this, the agent can infer direction and acceleration of the robot. This help as in prediction stage, for example to know that the prediction is telling to robot to go back.

1) Actor: The actor use a neural network to infer the best action to taken based on the state. The Figure 5 present the architecture used to actor.

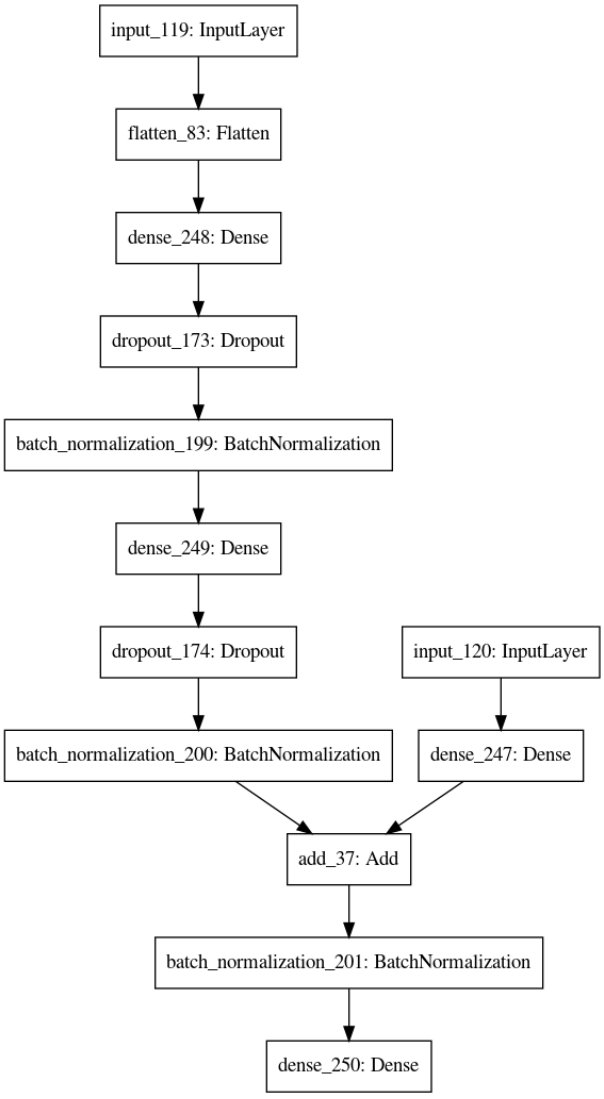
Fig. 5. Neural network architecture used by actor.



The dropout layers use a value of 0.4. The first dense layer has 512 neuros, and the second one has 32, all them use Relu as activation. The output layer has 5 neurons using softmax as activation. This represent the probability distribution of each possible action. The action taken by the agent is stochastic based on this distribution.

2) Critic: The critic use a neural network to infer how valuable is each action on the state given the state and action probability distribution. The Figure 6 present the architecture used to critic.

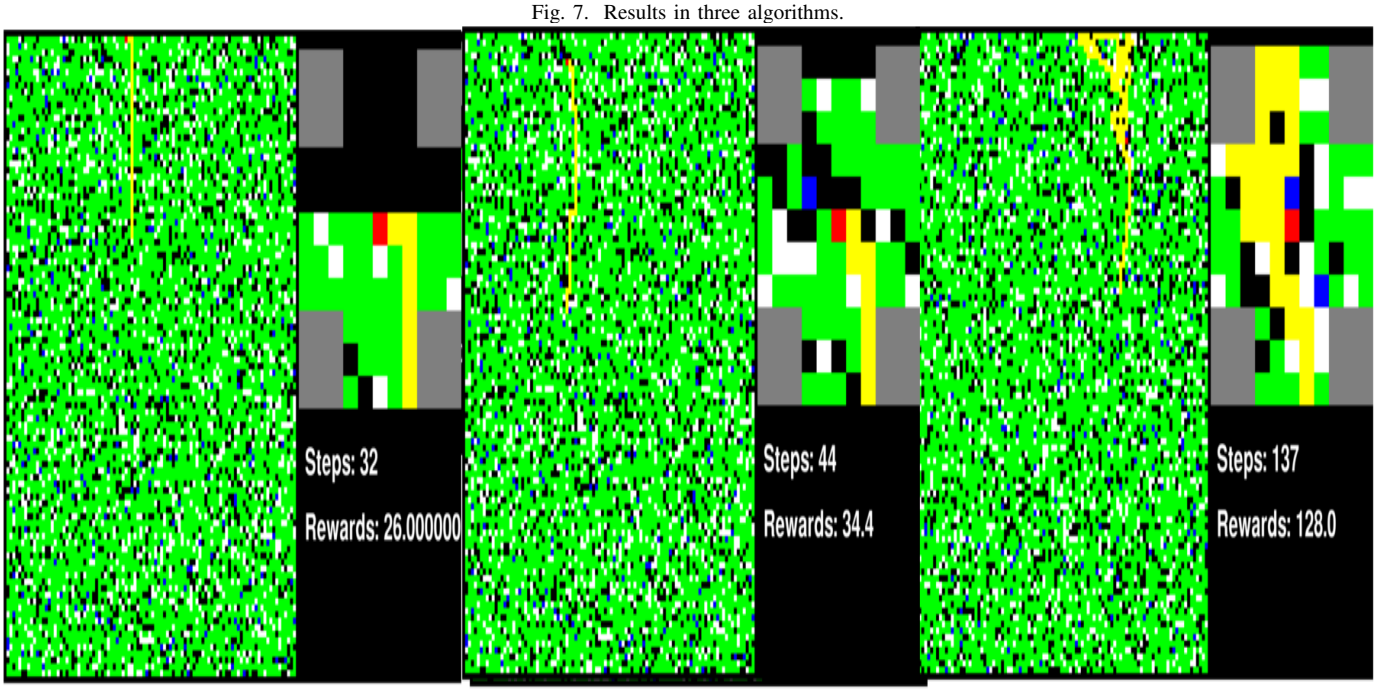
Fig. 6. Neural network architecture used by critic.



The dropout layers use a value of 0.4. The first dense layer has 512 neuros, and the second one and the dense connected in input_120 have 32, all them use Relu as activation, except by input_120, that use sigmoid to reduce the learning rate of the gradient in backpropagation. The output layer has 5 neurons without any activation. This represent the value predicted by the critic to this state with this action probability distribution.

3) Learning approach: Wait the epsode ends can leave the agent get a good total reward even if it takes some bad actions along its life. With this, we can fall in a optimal local

Fig. 7. Results in three algorithms.



solution. To avoid this, we use Time Difference Error with neural network estimation. So we can execute the train in each iteration.

In each step the following tuple is storaged on an agent memory:

(curr state, act prob, reward, state after action, end)

A batch of this memory is used after each step to improve the predictions of the agent in the following way:

The critic predict the value of *state after action* using the action probability predicted by actor in *state after action* and put it in a variable (*next q values*).

So the target to critic network follow Bellman Equation like as described in Equation 2 if the episode do not end or Equation 3 if it is the last step of the episode.

$$y = reward + gamma * next\_q\_values \tag{2}$$

$$y = reward \tag{3}$$

Then, the loss acquired by the critic looking to input 120 is applyed on actor to update its action prediction.

We used two identical neural networks to critic and the agent to improve the train stability, a model and model target. The second one is used to predict value of next states and has a learning rate much smaller.

V. Results

Tests was performed with three algorithms: Random Walk, REINFORCE and our agent. The source code can be downloaded to reproduce the results in Github [1] as well as the simulator.

The Figure 7 show the best result in each algorithm applying one hundred episodes. We can see that our algorithm clean much more region and get much more reward. So it is really clean the enviroument and not just travelling by the world.

Our algorithm was trainned by 8 hour in a GTX-960. When we try to increase the trainning time it start to be biased on one direction.

We try many networks architecture, for example, convolutional and deep fully connected neural networks. We think that for simplicity of the input data, theses complex networks tends be biased fast.

To test this, the learning rate was variable between 0.0001 to 0.00000001. When it is big, the networks start to show the same result to every state, making the agent just move in one direction.

Other actors handle this problem in different way, they try to cover the maximun area. So they always pass in cells that the robot can not clean.

VI. Future works

We understand that test just in simulator is not enough. For this we want to apply this in a real enviroument using cameras. With this more complex input data we believe that we could use more complex neural networks.

Another improvement in this work is the representation that the robot use to choose its action. In this work we use a simple images, but we believe that has better representations. The next steps is to create a graph representation of the world analyzing the images.

[1] https://github.com/evertonaleixo/smart_vacuum_cleanner

References

[1] Hongjun Zhou and Shigeyuki Sakane. Sensor planning for mobile robot localization—a hierarchical approach using a bayesian network and a particle filter. *IEEE Transactions on Robotics*, 24(2):481–487, 2008.

[2] Kazi Mahmud Hasan, Khondker Jahid Reza, et al. Path planning algorithm development for autonomous vacuum cleaner robots. In *2014 International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 1–6. IEEE, 2014.

[3] Iris Wieser, Alberto Viseras Ruiz, Martin Frassl, Michael Angermann, Joachim Mueller, and Michael Lichtenstern. Autonomous robotic slam-based indoor navigation for high resolution sampling with complete coverage. In *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*, pages 945–951. IEEE, 2014.

[4] Andreas Gylling and Emil Elmarsson. Improving robotic vacuum cleaners: Minimising the time needed for complete dust removal, 2018.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[6] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[7] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991.

[8] Mengfan Li, Chuanjiang Wang, Zhiqiang Chen, Xiao Lu, Meihua Wu, and Pengliang Hou. Path planning of mobile robot based on genetic algorithm and gene rearrangement. In *2017 Chinese Automation Congress (CAC)*, pages 6999–7004. IEEE, 2017.

[9] Zhongmin Wang and Zhu Bo. Coverage path planning for mobile robot based on genetic algorithm. In *2014 IEEE Workshop on Electronics, Computer and Applications*, pages 732–735. IEEE, 2014.

[10] Mohamed Amine Yakoubi and Mohamed Tayeb Laskri. The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of innovation in digital ecosystems*, 3(1):37–43, 2016.

[11] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[12] Sven Koenig, Reid Simmons, et al. Xavier: A robot navigation architecture based on partially observable markov decision process models. *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122, 1998.

[13] Peng Zhou, Zhong-min Wang, Zhen-nan Li, and Yang Li. Complete coverage path planning of mobile robot based on dynamic programming algorithm. In *2nd International Conference on Electronic & Mechanical Engineering and Information Technology*. Atlantis Press, 2012.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.