Application of Hopfield neural network on the energy-saving robot path finding and robot arm control.

Jérémie YANG Zhenyu 杨振宇
Student number : 19214064

September 3, 2020

Contents

Abstract

Robots are important tools in nuclear power plants since they could receive radiation without doing damage other than economy. It could be foreseen that once a material for robot fabricating that is radiation-resist enough, robot will be very universal in nuclear power plants. On the other hand, algorithms controlling the robot will be very important too. The complicated ground conditions in nuclear power plants require the robot to be able to climb stairs, avoid or step over various kinds of obstacles and make right decisions for saving energy.

Introduction

1.1 Literature review

1.2 Choice of technique

Traditional intelligent path finding algorithm using deep learning is an extremely hot topic and very well studied. In addition, they usually require a complete set of robot researching tools including a real robot which we do not possess. So we didn't choose to work with deep neural network but the Hopfield neuronal network. The principles and theory of Hopfield network are introduced in chapter 2. It is simple to setup and we've anticipated its potential additional functionalities. One is to use Hopfield's *diffusion factor* to take account of energy or time consumption. The other is to couple robot arm movement and space translation to globally control wheeled robots with arms. The two possible potential functionalities will be introduced in following two projects.

1.3 Projects to implement

1.3.1 Project 1 - 2D path finding problem

Project 1 consists of using Hopfield neuronal space to find an optimum path from an origin to a destination on a 2D map on considering energy and time consumption. Obstacles will be divided into 3 groups according the difficulty to pass them (in the sense of energy and time consumption). There are obstacles easy to pass, difficult to pass and impossible to pass. The robot will try to detour the obstacles but if the detour is slightly long, it will choose to pass the easy obstacles. If the detour is too long, it will choose to pass the difficult obstacles. Detailed description of the algorithm and results could be found at section 3.1 and 5.1.

1.3.2 Project 2 - Multi-arm target reaching problem

Project 2 consists of using Hopfield neuronal space to find an optimum configuration sequence for a robot with two-joint arm and XY dimension translation to reach a target in a 2D space. There will be obstacles that the robot should not touch and every point of the robot could reach the target (This is configurable). The robot could move freely on the plane and rotate its arms to which ever angle. This could help controlling robots that move with wheels and have

arms to perform certain tasks. Detailed description of the algorithm and results could be found at section 3.2 and 5.2.

Hopfield neutral space

2.1 Principles

Firstly, let us introduce the basic concepts of Hopfield neuronal space that are used in this project. Work space: The real space where the robot is working. The dimension of this space is usually two or three. Configuration space: A virtual finite space where every point is a configuration of the robot. Like the position or arm joint angle. The dimension of this space equals the freedom degree of the robot. (Note that the space can be continue or discrete, finite does not mean that there are countable number of points) Obstacles: In the work space and the configuration space, they are places that the robot cannot reach. Neuronal space: It is a discrete topologically ordered representation of the configuration space. Each point (which will be called "neuron") represents a configuration of the robot and the robot can go from every point to its adjacent point directly. Thus a path in the neuronal space will represent a feasible path for the robot. Neuron: Points of the neuronal space, each neuron is given a value between 0 and 1. The value will update in the finding process and finally indicate a path.

2.2 Advantages and disadvantages

Comparing with deep learning and neural network that are in plain development at present, Hopfield neuronal space has a very distinctive specialty –it does not require learning. Due to this, the run time performance is usually not as good as other neural networks.

Innovative algorithms to use for the two projects

3.1 Energy-saving path finding

3.2 Multi-arm robot

In this project, comparing to the first one, some additional steps must be taken. One is to calculate the robot's "presence" which means the space occupied in the work space by the robot's arms in each configuration. It is necessary for calculating the neuronal space. Obstacles includes, in addition to all points of the obstacles in XY dimension, all configurations that the robot's presence has overlapped space with any obstacles.

These calculations consume lots of computational resources. So various approaches that can reduce this consumption by thousands of times are proposed in the following chapters.

3.2.1 Robot presence calculation

The robot arms' special presence is calculated using two methods. One is to consider the distance between any point to the arm's joint and to the arm's central axis. The other is to use the equation of robots outer lines.

It is tested that the first one is faster and the result of second one is better. As we use a pre-calculation approach (introduced in section 4.3) to reduce the calculation time in this step there is no need to consider the computational resources consumed in this part. So we choose the second method.

Based on the joint coordinates, joint angles and arm size, the equations of the robot's arm's outer lines can be obtained.

3.2.2 Preparation of neuronal space

This preparation consists of calculating all the feasible configurations of the robot using the presences calculated. All neurons of configuration that has overlap with any obstacles are fixed to 0.

This takes fairly amount of time. So many computational approaches that accelerates the calculation are applied. The most powerful ones are the usage of Numba and the vectorization. (see chapter 4)

Optimizations of computer code

4.1 Vectorization

4.2 Numba and code modifications

4.2.1 Numba Library

Numba is a Python compiler that transforms Python code to high performance machine code which supports NumPy arrays and functions and loops. It could also work with CUDA to use Nvidia GPU.

4.2.2 Implementation of $np.roll()$ in Numba

4.3 Pre-calculation of robot's coordinates (Project 2 only)

Calculation of the robot's presence (spatial coordinates that the robot occupies) is rather resources expensive. Since the robot's move in space could be regraded as translation, it could be pre-calculated. Presences of the robot in different places could be obtained by simple addition of the presence of the robot of same arm configuration and the difference in the coordinates of their origin point. Let $\mathcal{R}$ denote the set of points that are occupied by the robot at origin point $(0, 0)$. Then the set of points occupied by the robot of same arm configurations at point (x_0, y_0) will be $\{(x + x_0, y + y_0)|(x, y) \in \mathcal{R}\}$. So at the beginning, we could directly calculate all the presences of the robot of all the arm configurations. Then no more robot presence calculation is required.

For a neuronal space of size 192×18, without pre-calculation, generation of neuronal space takes about : 44.5 s. With pre-calculation, generation of neuronal space takes about : 0.41 s, the preparation of coordinates takes about 0.14 s. An improvement of approximately $100\times$ can be expected from this method. Greater the space, more significant will be the improvement. Most importantly this method lowers the importance of the efficiency of the process of solving robot' s presence, which is in addition, not a simple subject.

Results and discussion

5.1 Energy-saving path finding

5.2 Multi-arm robot