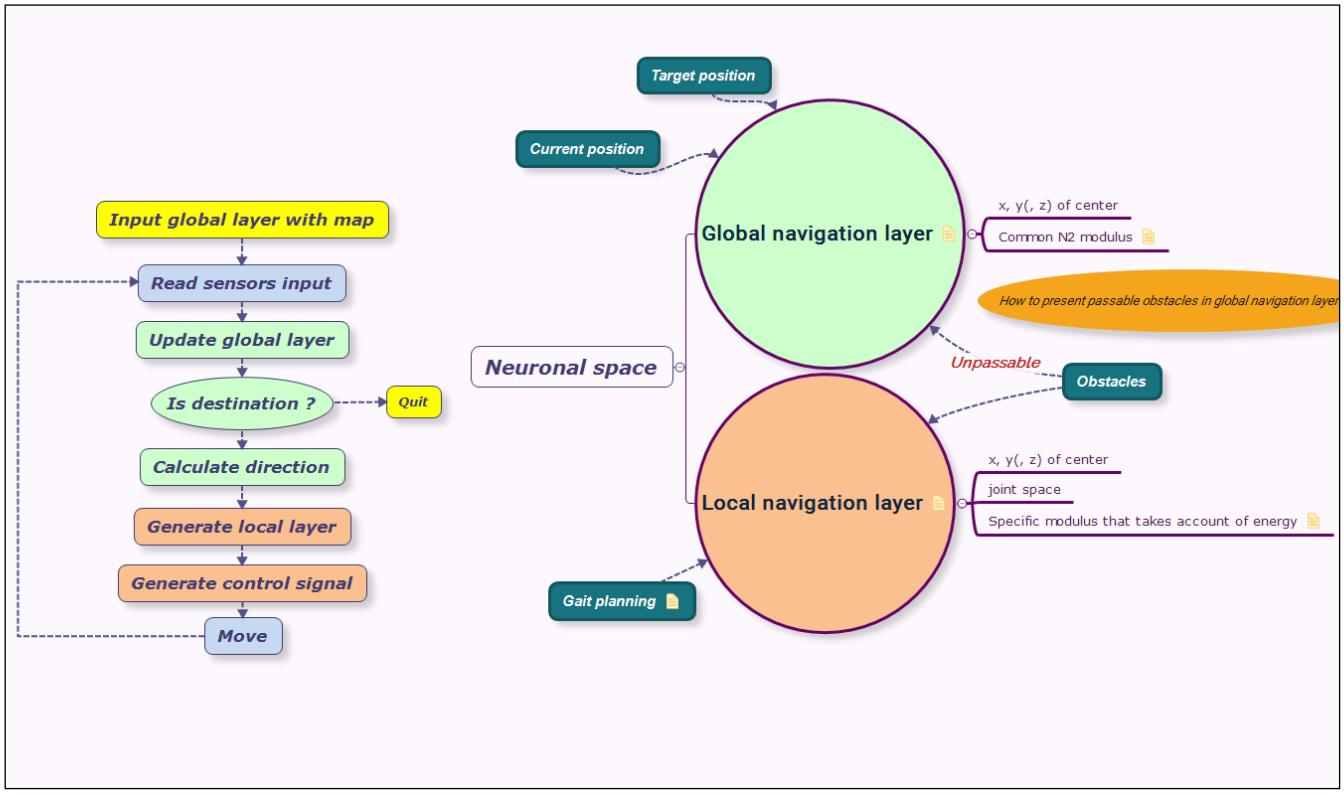
Hexapod robot control system

Year-end summary

J. YANG Zhenyu - January 9, 2020



First month - December 2019

The project started at the end of November of 2019, the first month was mainly about literature review. We've searched on Internet for papers about robot trajectory planning and obstacle avoidance.

What's good :

- Found an interesting paper containing the technique we've chosen afterwards
- Application of EndNote which is indeed a marvelous software for literature review

What's bad :

- Didn't choose the right platform to search, should have chosen SCI, lost fairly lots of time due to this.

After literature review, we've chosen a technique for the project - Hopfield network. Then we've started a simple test project in order to test the feasibility of this method. The simple project is easy to realize but covers the major part of necessary steps to carry out the final project.

Second month - January 2020

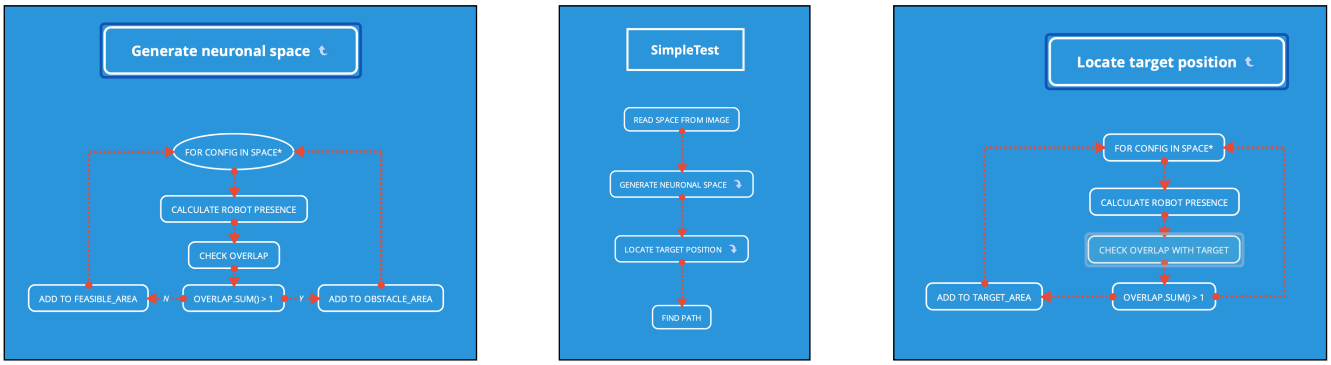
Simple project realization :

The simple project is a robot arm with 3 degrees of freedom. The robot moves in a 2D plane space. The objective is for the robot to touch a certain point in the space.

The project is written in Python3 or rather Anaconda3 with Cuda support in order to use GPU acceleration. After some research on Internet and several local test, usage of GPU will accelerate the project to an extent of 400x. (Using a test script in my PC with i5-9600F and Nvidia GTX 1060 6GB)

But due to my lack of Python programming experience and my habitudes of C programming, usage of loops if more frequently carried out rather than usage of vectors which is inevitable to use Python efficiently and use the GPU. This has to be corrected by vectorizing existing code. (There are serval difficulties anticipated)

The algorithms are shown below :



Algorithms in the simple project

Cuda - usage of GPU for computing :

Usage of GPU in Python code returns to the application of Cuda. Anaconda provides cudatoolkit library to make this process easier and prevent C programming.

My PC uses a Nvidia GTX which supports Cuda so the project can be accelerated , several test has been carried out. A 400x faster improvement is expected for this project with proper vectorization done according to a test script.

My laptop uses Inter and AMD gpu so this cannot be done in the winter holiday.

A new style of coding will be applied to accommodate this usage.

New method of calculating the presence of robot arm :

An idea came to me is that the coordinates of robot arm can be pre-calculated and stored locally to prevent calculation at run time. This could save a lot of calculation resources, with fixed robot arm angle configurations, the space that robot takes should also be fixed relative to its center point. (in this simple test project, at least 192x less calculation is needed) with this method, at run time (generation of neuronal space) will not demand any calculation more complete than addition and multiplication.

As for new method of calculating these coordinates, currently I've not yet found any promising methods. This seek will continue in winter holiday.