Controlling Devices by Using Robot Arm

By Jinsung Kwon and Jaekwang Lee

1. Introduction

We are willing to explore the possibilities to control several devices, such as mouse, keyboard, etc., which are around us, using robot arm. Among those devices, we apply machine learning theory in controlling the mouse. Mouse is a good example of devices which are hard to use control theories, because the relation ship between input (external force) and output (mouse pointer movement) is non-linear and difficult to define. For example, if we slide the mouse on the same side with the same force several times, the reactions of the cursor would be different according to the contact points.

To experiment this subject, we used PUMA560 robot arm in the Robot Manipulation Lab. We can read the mouse cursor movement and get a lot of movement data for machine learning by letting the robot arm touch the mouse in several possible ways and directions. We applied two learning methods, which are linear regression and reinforcement algorithm to move the mouse to desired position.

At section 2, we are going to introduce briefly about PUMA560 robot arm and our control interface. At section 3, we will compare linear regression and reinforcement algorithm in moving mouse cursor straightly. At section 4, we will use reinforcement algorithm in changing mouse cursor direction.

2. Controlling Robot Arm

PUMA560 is used for robot platform. The robot server is operating on a real-time OS, QNX. And we developed an Windows based client program (UI) by which a user can communicate with the server. This client program also loads MATLAB command windows and interfaces with MATLAB program. Consequently, our UI program communicates with the MATLAB program to calculate complicated computation and with the machine to move the robot arm.

Client	Server
- Request to move robot	- Control robot to move to desired position
- Request to stop robot	- Control robot to stand still
- Request to send current robot position	- Measure robot’s current configuration
- Measure the movement of the mouse	- Measure robot’s joint velocities
- Complicated Computation with MATLAB	- Measure forces and torques at the end effecter

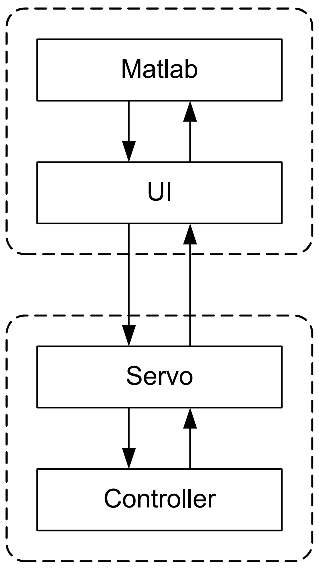

Figure 1. Robot Control Model

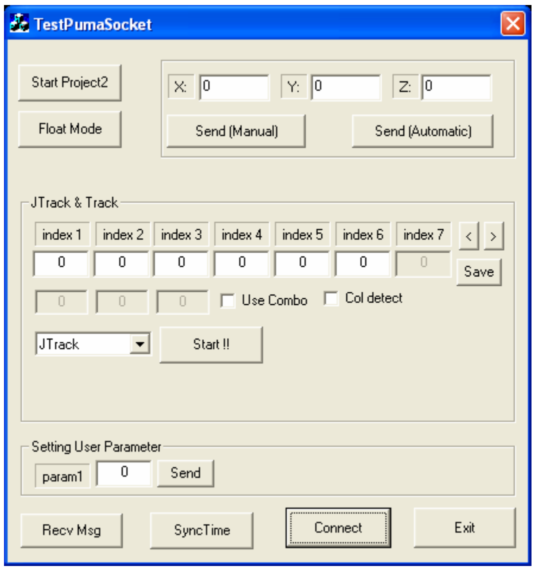

Figure 2. Client User Interface

3. Learning to move straightly (Linear Regression vs. Reinforcement)

As first experiment, we applied machine learning theory to move mouse pointer straightly. We applied two different learning algorithms and compared them. One algorithm is linear regression using least mean square and the other algorithm is reinforcement algorithm.

3.1. Linear regression

In linear regression algorithm, we trained machine by following way. At first, machine moves a little bit along x-axis, and then machine moves toward randomly chosen angle (-45º ~ 45º). Detailed figure is shown below. At each move, we gathered machine movement and mouse cursor movement, and then we tried to find the relationship with the mouse cursor position in Y-axis and those parameters.

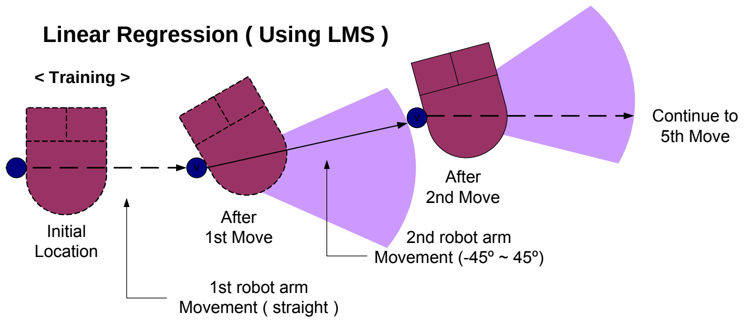
Linear Regression (Using LMS)
< Training >
Initial Location
After 1st Move
After 2nd Move
Continue to 5th Move
1st robot arm Movement (straight)
2nd robot arm Movement (-45º ~ 45º)

Figure 3. Linear Regression Training

Since there are many features (1st mouse cursor movement in x,y-axis, 2nd mouse cursor movement in x,y-axis, 2nd machine movement in x,y-axis and combined features of those primary features), we used feature selection algorithm to find "relevant" features. In testing each feature subsets, coefficients were found from randomly chosen 70% samples, and tested with the other 30% samples. After testing all possible features, we found those feature subset are most relevant to the result (mouse position in Y-axis after 2nd movement) which are,

1) Robot position in Y-axis after 1st move
2) Mouse cursor movement in X-axis during 1st move
3) Mouse cursor position in Y-axis after 1st move
4) Robot movement in Y-axis during 2nd move

3.2. Reinforcement

In reinforcement algorithm, we trained machine by following way. At first, machine moves a little bit along x-axis, and then machine determines its state. States are determined by following features.

1) Robot position in Y-axis
2) Mouse cursor movement in X-axis
3) Mouse cursor position in Y-axis

After finding current state, machine selects next action among 3 discrete directions and moves toward the direction. Detailed figure is shown below. A trial consists of 5 movements and if mouse position goes toward straight line (good action) during the trial, we gave positive reward. And if mouse position in Y-axis goes away more than 100 pixels from the base line at any time, we considered it as failure and restarted the trial.

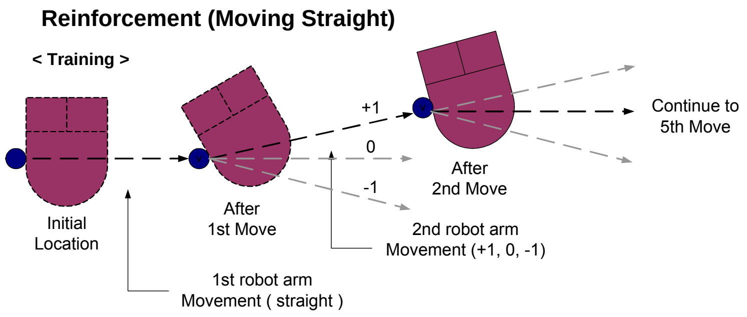
Reinforcement (Moving Straight)
< Training >
Initial Location
After 1st Move
+1
0
-1
After 2nd Move
Continue to 5th Move
2nd robot arm Movement (+1, 0, -1)
1st robot arm Movement (straight)

Figure 4. Reinforcement Training

3.3. Testing the algorithms

We tested both algorithms the same way as the reinforcement algorithm. Machine moves mouse 5 times consecutively and if the mouse position in the Y-axis is between the 100 pixels from the baseline, we considered the trial as success. On the other hand, if mouse position in Y-axis goes away more than 100 pixels from the base line at any move, we considered it as failure.

In testing linear regression algorithm, we chose different coefficients from 100, 200, 300, 400, 500 training samples and tested its performance. And in reinforcement algorithm, we did 20, 40, 60, 80, 100 trials and tested its performance. The result is shown below.

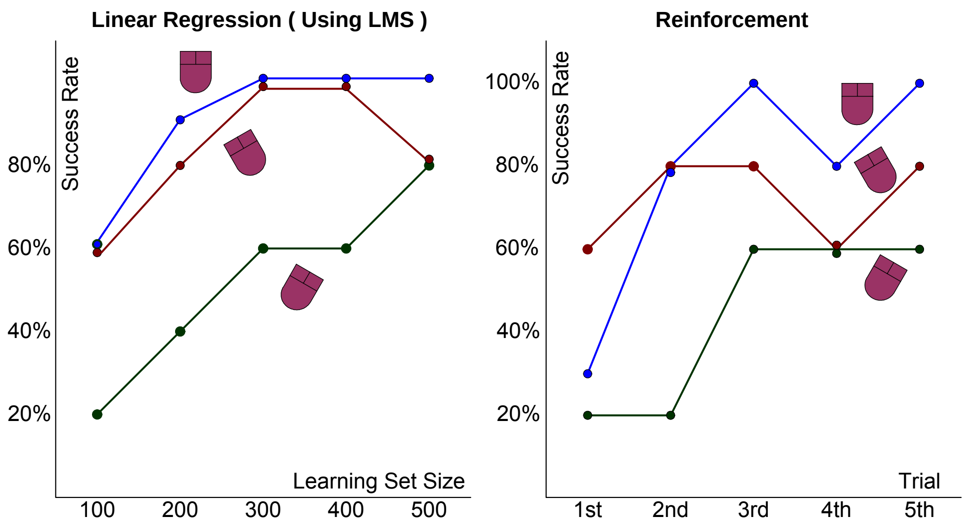

Figure 5. Test Result

3.4. Conclusion

After training and testing the machine using two different algorithms, we compared linear regression and reinforcement algorithm.

	Linear Regression	Reinforcement
Advantage	● Can make detailed target position both in X-axis and Y-axis ● Can take more various actions	● Fast learning with small number of failure ● Good performance even with complicated features
Disadvantage	● Slow learning with many randomized trials ● Hard to analyze the relationship between features	● Discrete actions and states ● State decision is very critical

4. Learning to Change Direction

At section 3, we tried to move the mouse straightly along X-axis. After moving along x-axis, we have to move the mouse along Y-axis to locate the mouse cursor at desired position. In this experiment, we are going to train the machine to change the direction correctly.

We trained machine by following way. At first, machine moves a little bit along x-axis, and then machine determines its state. States are determined by following features.

1) Mouse cursor movement in X-axis
2) Mouse cursor movement in Y-axis

After finding current state, machine selects next action among 5 discrete directions and moves toward the direction. Detailed figure is shown below. If 2nd mouse movement isn't along the right direction (vertical), test is considered as failure, and we gave negative reward to the trial.

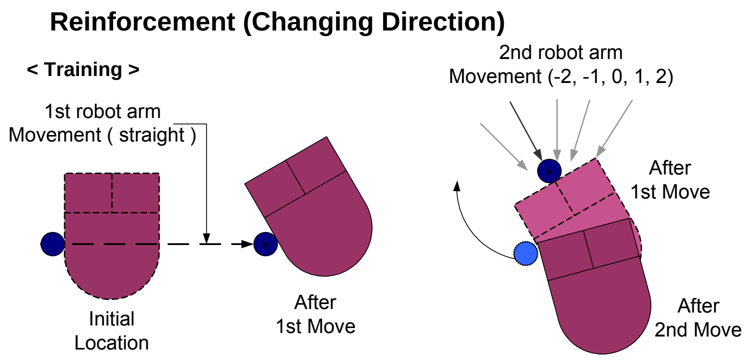

Figure 6. Reinforcement (Changing Direction)

Since it has more actions and states than the experiment in section 3, it took much longer time to train, but eventually we succeeded to change the direction.

5. Conclusion

If we combine 2 and 3, we can move the mouse cursor to desired position. By this, we can apply machine learning theories in controlling mouse without any complicated control theory or mouse position sensors. Moreover, if we use different shaped mouse, it's very difficult to find proper control rules, but we can use machine learning by applying this method to get the satisfactory result with any shape of mouse.