3D Mouse: Using Manipulator Motion As Input

Xiang Zhi Tan
xiangzht@andrew.cmu.edu

December 17, 2015

1 Introduction

One of the active research area in Human-Robot Interaction is the communication of intend between the robot and human user. Direct communication in this area are often done through voice commands or input devices such as keyboards and touch based interfaces mounted on the robot. Instead of relying on external system or additional hardware, we looked into how we could use an existing hardware on the robot, it's manipulator arm as an input devices.
One of the benefits of using the robot's arm is that we do not need to add additional hardware on the system. Furthermore, our method is less susceptible to factors in the environment like noises or light that might effect voice commands and computer vision techniques since our method relies on a direct physical connection that would not be influence by the surrounding.
As far as we know, none of the manipulators are designed with such functionality in mind nor systems been created to use existing arms as input device. Therefore, we designed a proof-of-concept system that shows it is possible to detect and classify user's intent through detecting different movement of the robot's arm by the user. Our system emulates how a normal joystick works in the real life and classifies the motions in high level gesture inspired action units like left, right, down and etc. In the following sections, we will describe the system design and preliminary test of the system.

2 System Design

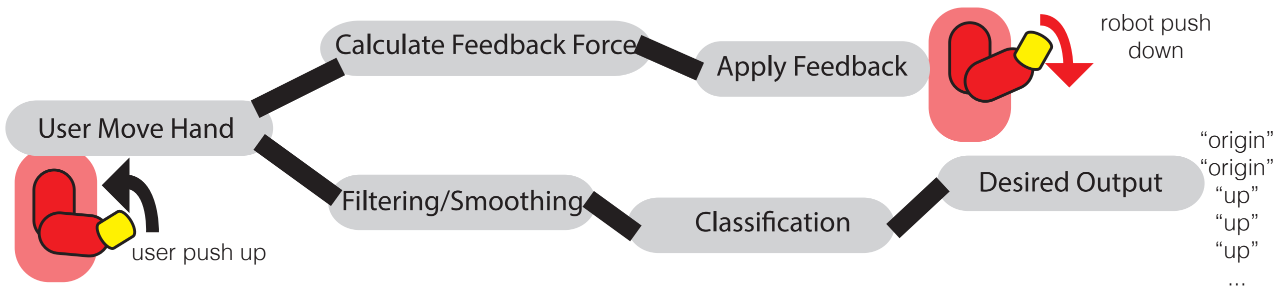

Figure 1: System flowchart shows how the the user's action(left) flows through our pipeline to calculate the required feedback force and infer the correct action.

The system mainly consist of two parts, a feedback system and an action classifier. A flowchart showing the system design is shown above in figure 2. The system was built for the Baxter Research Robot [1] that uses the Robot Operating System(ROS) [2]. However, we believe that the system can be easily extended to

[1]http://www.rethinkrobotics.com/baxter-research-robot/
[2]http://www.ros.org/

other robotic platforms with active compliant actuators. Baxter has two 7 degree of freedom arms with compliant actuators. Descriptions of both parts of the systems are as following:

2.1 Feedback System

The first part of the system simulates the physical property of a joystick where the arm will try to return to the resting position and apply greater force the further it is away from the resting position. We do this through attaching virtual springs at each joints. The springs are modeled using Hooke's law [4], which is the first-order approximation of the stress-strain relationship of a spring. Hooke's law is described as:

$$F = -kx \tag{1}$$

where k is the spring constant and x is the change in length of the spring. The relationship could also be described by the general stress and strain relationship

$$\sigma = Y\epsilon \tag{2}$$

Where σ is the stress, ϵ is the strain and Y is the Young's modules which describes the linear relationship between stress and strain. Our model does not model the yield point which is the point in the physical world where the relationship between stress and strain is no longer linear.

In our system, Hooke's law is applied on every single joint(7 per arm) on the robot where x is defined as the change of radian from the resting position and k was selected from trial and error. The selection of k for each joint was also influenced by movement restriction we wanted to imposed such as making actuators further away requiring more force to move. These restrictions where selected to make the classification process easier. However if we only applied Hooke's law, when the user release the hand, the arm will actually oscillate between points without any resistance(since the springs are virtually applied, real life physical resistance like air and loss of energy doesn't exist). To address this, a damping force [3] is applied to the system which is described with the following equation.

$$F = cv \tag{3}$$

where c is the damping coefficient and v is the velocity of the system. Through damping, we could also simulate a useful physical property of an oscillating system called **critically damped** which is the state where the system will return to the resting position as fast as possible without any oscillation. A critically damped system is a system where the damping ratio, ζ is equal to 1. The damping ratio of the system is calculated with

$$\zeta = \frac{c}{2\sqrt{mk}} \tag{4}$$

However, we do not know the actual mass of each individual component of the system, instead, the damping coefficient,c was calculated through trial and error until oscillation is no longer observable.

2.2 Classification

2.2.1 Filtering

Before classification methods were applied, we applied a common noise filtering algorithm, Exponentially Weighted Moving Average(EWMA) on the raw input data to smooth out the existing noise due to sensor errors. The filtering equation is defined as

$$x_t = \alpha y_{t-1} + (1-\alpha)x_{t-1} \tag{5}$$

where x_t is the filtered data at each time step, y_t is the raw input data at each time step and x_{t-1} is the filtered input at the previous time step. α is the control parameter that select which number to be weighted more. In our study, we selected α to be 0.25 through trial and error.

2.2.2 Feature Selection

We selected 14 different features that were used for the classifications. The features were the actual torque applied on each joint(one for each seven joints) and the position of the actuators in radian(one for each seven joints). The actual torque is different from the commanded torque in each command cycle as it also measures the torque that is applied on the system by external forces like the user pushing the arm in a certain direction. We choose not to use the end effector coordinates as a feature as we believe possible information gain in the 3D coordinate are already captured by our features.

2.2.3 Planned Output

We designed our system with the hope to distinguish 7 different actions that the user could perform, **upwards**, **downwards**, **left**, **right**, **forward**, **backwards** and finally **origin**.

2.2.4 Training Data

Training data for the classification was constructed with one member of the research team moving the robot arm with different actions and recording the values in the feature space at the end of each action. Each sample is then manually labeled with their respective actions by the same member.

2.2.5 Classification Algorithms

Because the way motion worked and the problem space, we believed the different actions are linearly separable to each other(they might be not linearly separable to all) in our feature space. We selected a few different classification methods that benefit from this property and compared their performance. We used the python machine learning library, scikit-learn [3] in our implementation of the system.

2.2.5.1 K-Nearest Neighbor Algorithm K Nearest Neighbor finds the K number of nearest point in the training data and picks the most common action in the nearest points with the distance as their weights. For our usage, the algorithm relies on using K-d tree structure, which is a binary tree where each node separates the high dimension space into 2 to optimize the search. In our study, K was set to 5 through trial and error.

2.2.5.2 Logistic Regression Logistic Regression is a linear model of the data and uses the following logistic function to calculate the probability for each action.

$$f(x) = log(1 + e^{-y_i w^T x_i}) \tag{6}$$

where x is the sample, y is the label and w is the weights of the linear model. The goal of the Logistic Regression is to find the weights that minimize the difference between logistic function and the training data labels. This can be expressed as the following minimization problem.

$$\min_{w} ||w||_1 + C \sum_{i=1}^{l} log(1 + e^{-y_i w^T x_i}) \tag{7}$$

Where C is the penalty of the error to prevent over fitting. The formulation and description of the used algorithm in the library can be find in [2].

[3]http://scikit-learn.org

2.2.5.3 Support Vector Machines The final method tried is Support Vector Machines with soft margins [1] that uses a linear kernel. The formulation of the problem is similar as we have done in homework 4. Where we tried to maximize the margin by solving the following constrained optimization problem:

$$\begin{aligned}
&\min_{w,b,\epsilon} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \epsilon_i \\
&\text{such that} \\
&y_i(w^T x_i + b) \leq 1 - \epsilon_i, \forall i \\
&\epsilon_i \geq 0, \forall i
\end{aligned} \tag{8}$$

Again, the algorithm we used solve the minimization problem through transforming the problem to the Lagrange function similarly to homework 4.

3 Evaluation

To evaluate the accuracy of each classification method, the system inferred possible actions with inputs from a pre-recoded test set. The test set was constructed with one member of the research team moving the arm around in an predetermined sequence that covers all the different actions. The actions at each time step is then labeled by the same member of the research team and set as the ground truth. We then feed the test data into the system and compared the inferred action with the ground truth. Following is the table that shows the accuracy of each classification method.

Method	Total	Correct	Error	Accuracy
K Nearest Neighbors	1437	735	702	0.511482
Logistic Regression	1437	1220	217	0.848991
SVM(Linear Kernel)	1437	1075	362	0.748086

Besides total accuracy, we also plotted out the comparison between ground truth and inferred action over all time step for each classification method.

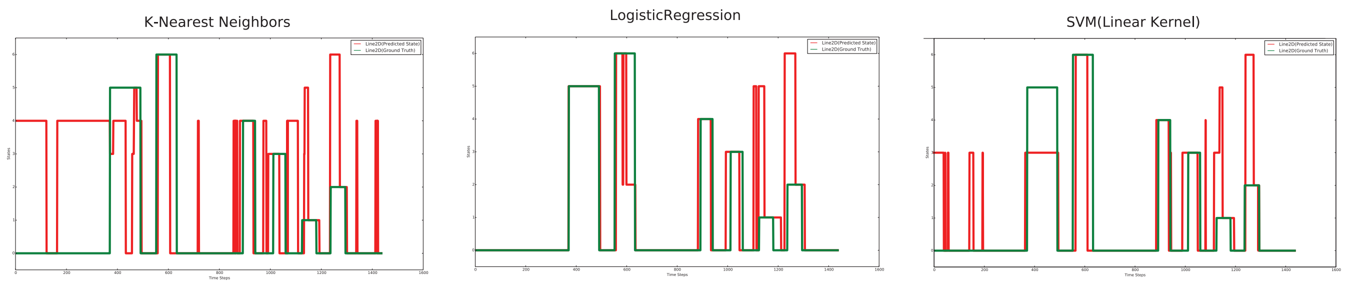

Figure 2: From left to right:KNN, Logistic Regression, SVM

The y axis is different types of action inferred by the system. The lowest point, 0 is the resting position(user did not move the hand). From bottom to up are forward, backward, left, right, upward and downward.

4 Discussion and Limitations

From the results , we observe that KNN performed the worst. One possible reason is that the training samples for action 4(the most common false positive) spans a larger feature space and overlap with over actions. This would have confused the system and often categorizing the action for that action. The next best forming classification is SVM. We hypothesized that SVM would perform the best and did not expect it to be outperform by Logistic Regression. Again looking at the data, the SVM had a false positive for action 3 more than others. SVM might have similar issues as KNN where their margin overlaps with others. Logistic Regression performed the best in our test set and also being the only classifier that was able to at some point classify all action correctly. However, there still are multiple false positive that needed to be address in future iterations. From a usability standpoint, the system might benefit more from having more false negatives instead of false positive since it will be more frustrating if the system misunderstood you compare to it doing nothing.

5 Limitation and Future Work

Currently, there are mainly two limitation in this study, the first would be the unreliability of the training data set. The training dataset only consist of one user's movement and about 5000 samples from that user. An increase in sample size and also variation by different users might increase the reliability and accuracy of the system.
The second limitation is our test set was only constructed from a same user from the test set with a very short duration. We only tested about 1500 samples which is equivalent to about 20 second worth of motion. A better test set would be constructed from multiple users that are sampled over a longer period of time, possibly about 5 minutes.
Future iteration of the system will aim to address these issues by expanding both the test and training dataset. While the system is able to predict and classify state at real-time, the test set is still pre-recorded. The next step would be migrate the current system to the robot and test the system in real life.

6 Conclusion

In this study, we shown a proof-of-concept system on how to repurpose the robot's manipulator arm as an input device. We described how our system emulates a joystick by modeling each joint as a spring around an resting position and classifying the user's intended arm action into 7 different states using three different classification method with Logistic Regression having an accuracy of about 85%. In conclusion, our system shows that it is feasible in using the robot's arm as an input device and we are excited at the possible applications that could benefit from this method of interaction.

7 References

[1] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[2] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research 9* (2008), 1871–1874.

[3] Wikipedia. Damping — wikipedia, the free encyclopedia, 2015. [Online; accessed 17-December-2015].

[4] Wikipedia. Hooke's law — wikipedia, the free encyclopedia, 2015. [Online; accessed 17-December-2015].