Correcting Mouse Input Error With Linear Regression

Bradley Gyemi
University Of Windsor Software Engineering
Tecumseh, Ontario
gyemib@uwindsor.ca

ABSTRACT

People who suffer from motor impairment disorders find it hard to interact with computers because of their inability to accurately use a mouse. Hand tremors and shaking caused from diseases like Parkinson's Disease make a computer mouse ineffective. My research proposes a system which uses machine learning to predict the intended endpoint of a user's mouse movement and correct input error which deviates from this predicted location. The results show that despite there being room for improvement, a linear regression algorithm is capable of predicting a user's intended endpoint with up to 75% accuracy and that the system can alter the users mouse position to a location closer to this intended direction. Though the system is capable of predicting intended location and correcting input error, its accuracy is reduced with more spastic inputs and the correction of error feels unnatural and forced for the user. In summary, the findings show that error can be predicted and corrected by an algorithm but the method of correcting such errors must be approached with a balance between meaningful adjustment, and delicate guidance of the cursor in mind.

1 INTRODUCTION

It is very important to be able to effectively use technology in this information age. Many people have difficulty accurately using computers due to various health conditions like Parkinson's and Huntington's disease which cause involuntary movement. My research seeks for a way to use machine learning to determine to which degree a mouse input is voluntary and to which degree involuntary movements can be corrected to make computers more accessible to users with motor impairments. This research could prove useful to anyone who is looking to use machine learning to learn from user mouse input and make predictions based on the mouse data available to them. The objective of my research is to design a program which is capable of learning from the user's mouse movements and using this knowledge to alter their mouse input when an involuntary movement is detected to effectively "filter out" shaking and spasms from their input. In the case of users without a motor control disease inhibiting their movement, small errors are still present in their input so this technology could also be used to maximize precision during mouse movements.

2 BACKGROUND SURVEY

Other studies on mouse input have been done in an attempt to garner information from user mouse input. Many of these studies are done with the purpose of classifying the user's movement in order to identify/authenticate them [2], while others attempt to use the information to attempt to gain information about the user's mood, motivations, age, gender, etc.. The mouse information in all cases is broken down into data relating to the direction, speed, and distance travelled (other information is determined through these measurements) by the mouse pointer, as well as any mouse related events which are triggered in the time of the movement like clicks or spins of the scroll wheel.

I intend to use this information to try and predict the endpoint of the mouse movement, then assist the user in getting to that endpoint. Endpoint prediction is done most recently by Pasqual and Wobbrock [4] in 2017 through the use of a kinematic template stored from the user, and application of that template on future data given by that user. This kinematic template uses properties of kinematic movement like acceleration and jerk to potentially circumvent or "skip" steps in the process of moving the mouse from one location to another. Their algorithm could predict the endpoint of the user's movement with 70% accuracy using prior mouse input data. Inverse optimal control is another method of predicting mouse endpoints used by Ziebart, B., Dey, A., Bagnell in their study [5]. This method breaks mouse movements down into continuous states, then uses a probabilistic control technique to observe pointing trajectories given the known pointing target, and finally it uses Bayes' rule to assign probabilities of the intended targets based on components of the cursors trajectory.

2.1 Machine Learning and Linear Regression

The method I chose uses a form of machine learning to try and build a profile of a given user's movements when their target is already known. This was chosen with people with motor impairment in mind because these deterministic kinematic approaches are unlikely to work when the input is so shaky and inconsistent. For example, it would be very difficult for an algorithm which only calculates based on kinematic angles observed to predict an endpoint to the left of the mouse cursor when the user is experiencing an involuntary hand shake towards the right. Machine learning, at the highest level, is basically the processing of input to output using a function which changes the way it processes future data based on the results of data that it previously processed. In this way, the machine "learns" from processing more data so it can more accurately process future data. I am hoping that my algorithm will learn from the hand shakes that it knows are not toward a target which it is given so that when it isn't given a target, it can determine the target's location despite these involuntary hand movements.

One method of machine learning which is based on creating lines of best fit through data is linear regression. Linear regression is mentioned as an option in all studies which I have referenced but is not chosen by any as a method of predicting location. Since this study seeks to both predict intended location and correct error, it makes sense for me to use a method that is as intuitive and accessible as possible while still being effective. This is the reason why linear regression was chosen for my method of predicting mouse endpoints. Linear regression is a method which uses a large

amount of data to calculate the relationship between values which will always be known (attributes) and values which we will try to predict later on (labels). These relationships are determined by plotting data points on a multidimensional graph and calculating a line of best fit formula for a line that best moves through these data points. When it is time to predict the value of unknown labels based on known attributes, this line of best fit formula will be used to calculate the value of the labels by inputting the attributes into a y = mx + b style line as the x value. Since there are more than one "x" value and more than one "y" value (exists in multidimensional space), the equation of the line determined by the algorithm looks as follows instead:

$$y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \varepsilon, for\ i = 1, 2, ..., n \quad (1)$$

The epsilon represents the error (deviation) which the linear regression algorithm tries to minimize by optimizing it's line formula. It does this by assigning appropriate weights (represented by beta) for parameters 1..n. The stronger the parameter's linear relation to the label y, the greater its weight. The model used in the case of the library used in my algorithm is the least squares model. The method of this model is best described by an item from Yale found at [6], "the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line".

Given the recent growth of machine learning and it's increased accessibility to programmers, I decided to use linear regression to try and determine the endpoint that the user was aiming for. This method has been used before but not with parameters that I selected to input into my linear regression model. Some of these values put into the model were derived from the kinematics template from [4], some others from [5], and others were my choice.

3 MY APPROACH

Correcting a person's involuntary mouse movements using a machine learning algorithm was done in 6 steps:

(1) Collect mouse input data from the user in such a way that the user's intended target is known and recorded with each frame of mouse data.
(2) Select a model to process the user's mouse input data with (linear regression was the chosen model).
(3) Train the model using the mouse input collected in step 1
(4) Repeatedly train models using the data set supplied by the user and save the most accurate one to a retrievable file
(5) Have the user provide mouse input again, this time feed the user's input through the model each frame to predict where the user is trying to aim the mouse.
(6) Affect the user's inputs so they more closely follow the most efficient path to that predicted intended location.

Firstly I built a MouseLogger class which has methods capable of logging mouse data, refactoring it to give to the linear regression model, and altering mouse cursor location based on suggestions from the model. This is the main class which is responsible for all of the computational logic outside of predicting the target's intended location (handled by another class which soon to be explained). Here are the main computational processes handled by the MouseLogger:

3.1 MouseLogger Function 1: GetAverageAngle()

Traverses through a linked list of items containing the 20 most recent mouse data entries (frames) and averages the angle direction of the mouse movements over the 20 most recent frames. This is done by converting all recent mouse vectors to their corresponding unit vectors and adding them together tip to tail using vector addition.

Convert vector r to its unit vector

$$\vec{r} = (x, y) \quad (2)$$

$$u(\vec{r}) = (\frac{x}{||r||}, f\frac{y}{||r||}) \quad (3)$$

Taking the average of the 20 most recently observed angles in a set of vectors R of length n is adding the unit vectors tip to tail with simple vector addition. This approach allows this metric to avoid over-evaluating an input due to a disproportionately fast movement (likely caused by a spasm)

$$avgAngleVector = \sum_{k=n-20}^{n} u(R[k]) \quad (4)$$

Returning the direction of the avgAngleVector returns the avg angle observed over the 20 most recent frames of data.

3.2 MouseLogger Function 2: GetAverageSpeed()

Takes the average of the 20 most recent speed values by traversing through a linked list of the items containing the 20 most recent mouse speeds observed in the 20 most recent frames. Let S be the set containing the observed speed at each frame.

$$avgSpeed = \frac{\sum_{k=n-20}^{n} S[k]}{20} \quad (5)$$

3.3 MouseLogger Function 3: generateDetailedCSV()

Takes in the basic data from logMouse and uses it to derive additional data about the mouse movements to eventually be used to train the ML model. Let the current cursor position be the nth point in a set of logged mouse data points M of length n. Let the cursor's target position be the nth point in the set of logged target positions T of length n. A point in either of these sets is of the form (x,y). Let the time passed since the system start time at the nth frame be the nth point in a set S of length n (measured in milliseconds). The additional data is defined as follows.

$$IntendedDirection = \arctan(\frac{T[n][1] - M[n][1]}{T[n][0] - M[n][0]}) \quad (6)$$

$$TargetDistance = \sqrt{(T[n][0] - M[n][0])^2 + (T[n][1] - M[n][1])^2} \quad (7)$$

$$ObservedDistance = \sqrt{(M[n][0] - M[n-1][0])^2 + (M[n][1] - M[n-1][1])^2} \quad (8)$$

$$ObservedSpeed = \frac{ObservedDistance}{S[n] - S[n-1]} \quad (9)$$

$$ObservedAcceleration = \frac{ObservedSpeed}{S[n] - S[n-1]} \quad (10)$$

Table 1: Software Tools and Their Role in The System

Software Tool	Role In System
Homebrew	Package manager
Python	Programming language of choice
Pygame Library	User interface and frame by frame event loop
Pyautogui Library	Recording mouse input and changing cursor location
Pandas and csv Libraries	Storing and retrieving mouse input in files
Numpy Library	Refactoring and querying stored data
sklearn Library	Creating and training linear regression model to predict intended target
pickle Library	Storing and retrieving machine learning models
matplotlib	Graphing relationships and findings

Table 2: Hardware Tools and Their Role in The System

Hardware Tool	Role In System
Macbook Pro running MacOSX	Machine of choice for development and testing
Razer Death Adder Mouse	Input Device

$$ObservedDirection = \arctan\left(\frac{M[n][1] - M[n-1][1]}{M[n][0] - M[n-1][0]}\right) \quad (11)$$

For the following 2 metrics, let V be the set of vectors of length n-1 generated with a magnitude equal to ObservedDistance and a direction equal to ObservedDirection (given in radians from $\pi -> -\pi$ through the python math.atan2() function). Let S be the set of length n containing the ObservedSpeed value at each frame.

$$RecentAverageDirection = avgAngle(V) \quad (12)$$

$$RecentAverageSpeed = avgSpeed(S) \quad (13)$$

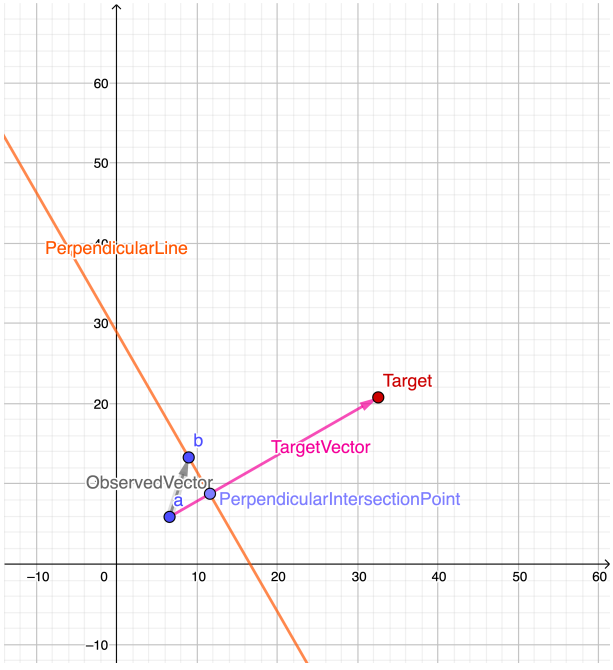
3.4 Predicting Intended Target Location with LinearRegrssionAlg

The LinearRegressionAlgorithm class is responsible for training the linear model and using that linear model to predict the values of IntDirect(n) and TargetDistance(n) when the target location is not known. It must contain functions which take in mouse data stored in csv files. It must use this data to train a linear regression model to predict the intended mouse direction and target distance from the cursor. It must test the accuracy of the linear model generated and repeat this process a sufficient number of times to choose the model with the best accuracy. It must then save this model to a ".pickle" file to be later retrieved and used by the MouseLogger to predict the distance and direction to the target and alter mouse data accordingly.

3.5 Altering Cursor Location

Once the system is ready to predict the intended target's direction and distance from the cursor, it is ready to make alterations to the user's input based on this information. The alteration is determined by finding the distance between the observed cursor location (b in Figure 1) and the line which connects the cursor location at the previous frame (a in Figure 1) and the predicted target location at the previous frame (TargetVector in Figure 1). The altered cursor location is then set somewhere along the line which runs perpendicular to the TargetVector (PerpendicularLine in Figure 1) connecting that line to the current cursor location (b in Figure 1). The new point goes between the current cursor location (b) and the intersection point of these two lines (PerpendicularIntersectionPoint in Figure 1) as shown below in Figure 1.

Figure 1: How Altered Location is Generated



3.6 Wraping Up the System

Once the logic is built the next step is to design the order of operations of the main module located in the file called "PygameWindow.py". Information in this system is processed by order of a Pygame event loop in this file which updates at 60 Hz and oversees the activity of the active class objects in the system. They include the various screen managers which present different interfaces to the user at different stages of the experiment and the MouseLogger which logs and alters mouse data. Instances of these classes are created and their respective setup functions are run before the event loop begins to minimize processing during the loop.

Lastly I built the classes and methods which are responsible for managing both the presentation of the visual objects on screen and the flow of actions performed by the Mouse Logger and the linear regression algorithm. The system states are organized into "screens" or displays with instructions on the screen for the user to follow. The classes designed to manage the screens were stored in a file called "PygameTrainingKit.py" and were created 1 for each screen (or interface you could say).

4 EXPERIMENTS

The system was tested with 2 different users, one of which had parkinsons which caused involuntary shaking of the hands, and the other had no motor impairments. They both were put through the experiment twice; once to test how well they liked using the mouse with the mouse correction taking place, and another time to instead have their mouse position not altered but to graph how their mouse would have travelled had the corrective algorithm been active.

The system walks the user through the experiment by guiding them through a series of screens as follows

4.1 Experiment Breakdown

- Opening Screen: user inputs their name so that all of their data is stored in files which are named after this input.
- Pre-Training Screen: user is prompted with instructions and a start button which, when clicked, takes the user to the Training Screen.
- Training Screen: The user is presented with a blue dot on the screen (representing a target) and a line drawn from the location of the previous target dot to the current target dot. As the user moves their mouse along this line toward the target dot, their mouse location, timestamp and the location of the target dot is recorded at every frame at a frequency of 60Hz. When the user reaches the dot and clicks on it, another dot is drawn elsewhere on the screen and a line is drawn connecting the location of the mouse and this new target dot. This process continues until a sufficient amount of data is collected. When the final dot in the training sequence is clicked, the user's raw mouse data is sent to another class called the MouseLogger which saves the data into a csv file called "*user_name_basic.csv*". The MouseLogger uses that data to derive the detailed data in figure 3.0 and saves it in a csv file called "*user_name_advanced.csv*".
- Pre-Testing Screen: The user is prompted with instructions and a start button which, when clicked, takes the user to the Testing Screen
- Testing Screen: The linear model for the user which is saved in "*user_name_advanced.pickle*" is loaded back into the MouseLogger object. The user is given the same type of task he/she was given at the testing screen though target dots and target lines. The difference is that this time the data associated with the location of the target dot is not given to the algorithm. As the user moves along the target line toward the target dot at frame n, he/she has his/her input entered into their unique linear model which predicts a line from their mouse cursor at frame n to where it thinks the target dot is. The cursor location is then altered by the MouseLogger. How well this system feels to use for the user will tell me how effective it is. In order to have more observable quantifiable data, the algorithm may be set to use this predictive data to draw two lines each frame instead of altering the mouse's location. One line represents the actual, unfiltered mouse movement, and the other is drawn in the location that the mouse would have moved had it been altered by the system. Observing the behavior of these two lines may also give insight to how well the system is predicting intended location and altering the mouse cursor to better reach that location.

4.2 Results

First were the results from the user without any motor impairment. This user observed an average of a 71% accuracy over 5 tests with results ranging from 65% to 75%. The algorithm was able to predict the intended direction of the user with an even higher accuracy though the distance away from the cursor was harder for the model to pin down. The user's impression of the software was overall negative. The user complained that the software was too forceful in changing their inputs which resulted in changes to the input which would have been easier to fix by readjusting than being forced back into the proper position. In the testing version where corrective lines were drawn for each frame however, 70% of adjustments were more optimal than the observed mouse movement. This shows that though the algorithm is working, it does not feel good to use and it does not alter the user's input in a way that makes them satisfied.

The results from the user with Parkinson's Disease were also unconvincing. This user observed an average of 51% accuracy over 5 tests with the results ranging from 46% to 55%. The algorithm was able to predict the intended direction of the user with only a 49% accuracy. The user's impression of the software was overall moderate. Since the user is used to not being able to use a mouse well anyway, the user appreciated the corrections that were made on involuntary movements which would have had massive impact on the user's input. However in addition to these large corrections came a plethora of errant alterations to the user's input. In the testing version where corrective lines were drawn for each frame, the algorithm made accurate corrections 60% of the time which was surprisingly high given the model's predictive accuracy. Though the algorithm failed to detect many errors and also created errors which weren't input ed, the corrective line was much less erratic than the original input. The largest and most obvious errors were usually corrected which overall, despite performing more poorly than with the user without motor impairment, resulted in the system being better received by the user with Parkinson's Disease.

4.3 Observing Strongest Data Relationships

Figure 2: Intended Direction vs Recent Average Error

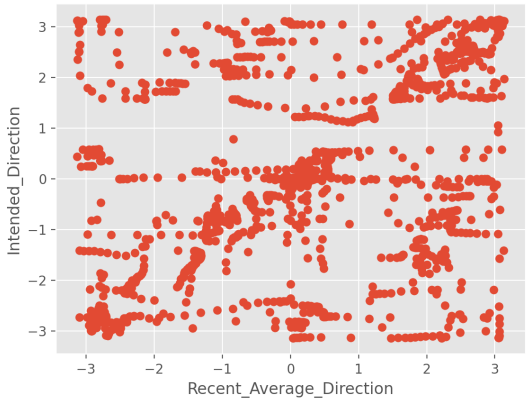
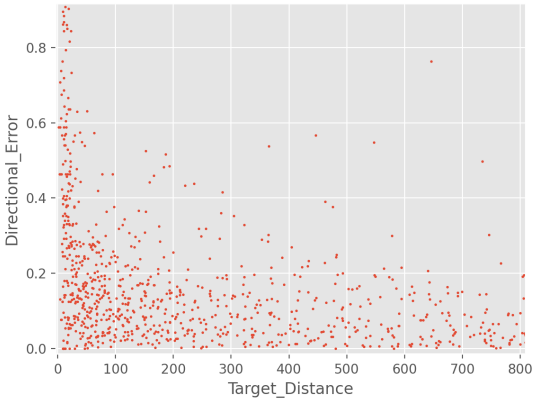


Figure 3: Directional Error vs Target Distance



5 DISCUSSION

The results show that using the data which I observed in the training phase allows a linear regression algorithm to predict, with adequate accuracy, where the user is intending to go with the mouse. The algorithm can make adjustments that bring the user's mouse closer to the most direct line to the target location. This was the role of the algorithm so one could say that the results were somewhat of a success. My work shows a way that a mouse input error can be identified and corrected, however the method of correction which I used does not feel natural for the user because it changes the observed direction of the cursor that the user sees on screen. This feels especially unnatural when the system inaccurately predicts the user's intended direction. Having one's input taken over by an algorithm is a delicate balance between guiding the user's input without forcing it to a place that makes the user feel like their input is not driving the cursor's location to a high enough degree.

Figure 4: Recent Average Speed vs Target Distance

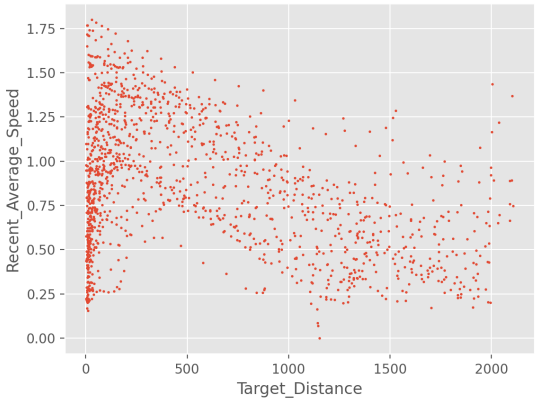
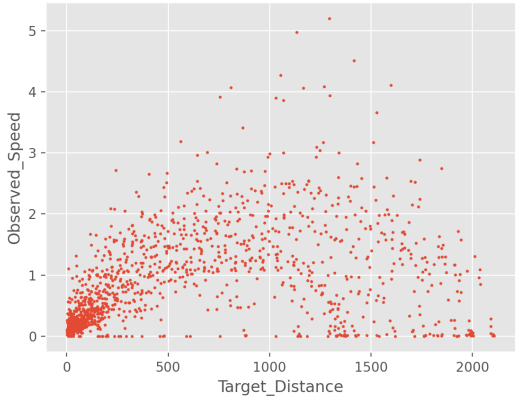


Figure 5: Target Distance vs Observed Speed



5.1 Disadvantages

Despite the system doing what it is intended to do, it does not solve the problem well enough to solve the issues faced by motor impaired users. There is an issue with the accuracy of the algorithm which ranges from 65-75% for a user without any motor impairment but drops to 45-55% for users with motor impairment. Since the objective was to find a solution which is directed toward users with motor impairment, this level of accuracy is not even close to being high enough to accomplish the goal which I set out to accomplish. Therefore the algorithm is not accurate enough especially when the input is unpredictable and spastic which is to be expected from a user with motor impairment.

Additionally, the algorithm goes about mouse correction by forcibly changing the location of the mouse at every frame to a different location. This, in theory, brings the mouse closer to the path it needs to follow to get to the intended target. However in practice, having one's mouse movements forcibly altered to a different direction and a different location feels awkward and unnatural.

Both users report a difficulty using the mouse when the alterations were active even when those alterations were "correct" in the sense that they brought the cursor to a more optimal location. This is because even though there is a slight error in the user's input, the user can still sense this error in their movement and hence he/she automatically adjusts later on in the mouse's trajectory. But with the algorithm active, these adjustments result in another error in the other direction instead of a correction of the previous error.

5.2 Advantages

Despite much room for improvement, the system works. It trains itself on the unique user's movements, uses that data to predict their intended location at a respectable accuracy, and it adjusts the user's input when they deviate from the direct line to that location. This clarifies a step by step process in using linear regression to correct mouse input error and as such it brings value to the search for finding an applicable solution to the problem that users with motor impairments face when using a mouse. The system is user specific, effective, and efficient considering the amount of data it has to store and process under a time crunch. Even though this solution is not applicable to the problem due to its shortcomings in accuracy and "natural-feeling" error adjustments, my research brings forward a solution that has not been attempted in quite the same way before. The use of a perpendicular line drawn against a predictive line connecting it to the observed pointer location can be useful in other scenarios where the user's comfort of use is not a large priority; for example a software used to aim pointers without user input.

5.3 Limitations

Some limitations of the software which are due to the use of machine learning is that it must be trained in order to function. The user can't just run the software and continue using their computer as they were. The user must first spend some time to train the algorithm in their own mouse habits using the software's training regimen because otherwise it won't have the known target location values with which to build the linear regression model. Another limitation of the software is that it requires the user's computer to be able to process the adequate amount of mathematical logic for detection, prediction, and correction in every frame. This is a significant amount of processing to do in a fraction of a second which is why decreasing the amount of mathematical operations per frame was a large focus on the development of the system. Even with this focus, the user's machine must be powerful enough to run the software. The machine used to run the experiments has a powerful processor and plenty of memory and not all users will have a machine quite so capable.

5.4 Evaluation

I did not meet my goals/objectives. The software fails to solve the issues experienced by motor impaired users with regards to mouse input. The software solves these issues 50% of the time and creates new issues the other 50% of the time. As such it is not ready to be used to help these people get smooth access to computers. Though the system has given me and hopefully those who read this report new knowledge and a new approach to thinking about these issues, it cannot be denied that this system is not ready to solve the problem that I set out to have it solve.

6 CONCLUSION

The system I proposed in this report seeks to assist in making computers more accessible to those users with motor impairments which cause involuntary shaking of the hand. At the highest level it accomplishes this in 4 steps. The first step is to train a user specific linear regression model through a training process which has the user click on targets who's locations are known by the algorithm while the user's mouse input is logged. Secondly, the system repeats the generation of this linear regression model and selects the most accurate instance of it to use in the third step. The third step is to take in the user's mouse data every frame to be processed by the linear regression model selected in the second step and use it to predict the user's intended direction at the current frame. The fourth step is to observe the distance that the user's cursor has deviated from the prediction line generated in the third step and alter the cursor's location to be closer to that line. The system generated accomplishes this task effectively at 65-75% with normal users and 46-55% for users with mother impairment. Additionally the correction of the mouse location was reported to feel forced and unnatural for users who tested it. As such the research is good but not good enough to make computers more accessible to users with motor impairment.

6.1 Future Work

The first step to improving this research is increasing the accuracy of the model used to predict the user's intended location. If one intends to alter a user's mouse inputs, he/she better be making correct alterations almost all of the time. Additionally the system can be improved by changing the way the corrections are carried out. A perpendicular line covers a range of angles that the user did not actually input which I believe contributes greatly to the unnatural feeling observed in the experiments. Perhaps reducing the magnitude of directional change or reducing only the distance travelled during inputs that are perceived to be errant would be more acceptable to users. My hope is that this research brings some ideas to the minds of those who seek to further develop technology meant to make computers more accessible to those with motor impairment.

ACKNOWLEDGMENTS

This work was supported by Dr. Ziad Kobti. Thank you for your patience and guidance.

7 APPENDIX

7.1 Code

The code for the algorithm is located in the following github repository: *https://github.com/BRADiical/LinearRegressionMouseCorrection*

REFERENCES

[1] Hwang, F., Keates, S., Langdon, P., and Clarkson, J. (2003). Mouse movements of motion-impaired users: A submovement analysis. ACM SIGACCESS Accessibility and Computing, (77-78), 102. doi:10.1145/1029014.1028649

[2] Pusara, M., and Brodley, C. E. (2004). User re-authentication via mouse movements. Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security - VizSEC/DMSEC '04. doi:10.1145/1029208.1029210

[3] Trewin, S. (1996). A study of input device manipulation difficulties. Proceedings of the Second Annual ACM Conference on Assistive Technologies - Assets '96. doi:10.1145/228347.228351

[4] Pasqual, Phillip and Wobbrock, Jacob. (2014). Mouse pointing endpoint prediction using kinematic template matching. Conference on Human Factors in Computing Systems - Proceedings. 10.1145/2556288.2557406.

[5] Ziebart, B., Dey, A., and Bagnell, J. A. (2012). Probabilistic pointing target prediction via inverse optimal control. Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces - IUI '12. doi:10.1145/2166966.2166968