Determining Manipulation Task Difficulty Based on User Input Frequency

Introduction

Robots and robot-human teams have the potential to significantly improve the quality of life of those with disabilities. Presently, significant research is underway to define methods to afford users of assistive machines the opportunity to safely retain as much control of their machine as possible. One solution to this goal lies in dynamic autonomy allocation. To inform an assistive machine when to allocate or retain user control, a variety of metrics may be investigated. In this project, the frequency of a user's input to control a robot to perform a manipulation was investigated as a possible metric for informing an autonomy allocation arbiter.

The primary goal *of this project is to test feasibility. That is, to process the user input data and implement various machine learning methods to test if user input frequency is a feasible standalone metric for task difficulty and to inform future research.*

Data

The data used in this project is the result of [1]. In [1], 20 able-bodied individuals completed a series of 13 manipulation tasks two times each as to determine which kinematic features (x-, y-, z-translations and rotations) contributed to a task's difficulty. Each task was completed with and without assistance; assistance followed a blending paradigm of 50%. The user's raw, timestamped joystick inputs were recorded for all tasks. At the conclusion of each task, a user would complete a survey to classify perceived task difficulty. The survey completed modeled the NASA TLX (task-load index) [2]. It has been shown that the NASA TLX score is a good metric for classifying task difficulty [3]. A higher TLX score indicates a more difficult task. This generated a dataset of 520 labeled, timeseries joystick examples.

To preprocess the user input frequency data, the data was extracted from its raw format (*.bag file, used in ROS, the Robotic Operating System) to *.csv. Then, at the advice of a presently in-review paper from the author's lab[1], a 10-point moving average filter was applied to the data. Figure 1 presents an image of the user input frequency data after application of the 10-point moving average filter. This set will be entitled 'primary.' A second dataset was produced where two consecutive 10-point moving averages filters were applied. This was done to study the effect of a second filter; this set will be entitled 'secondary.'

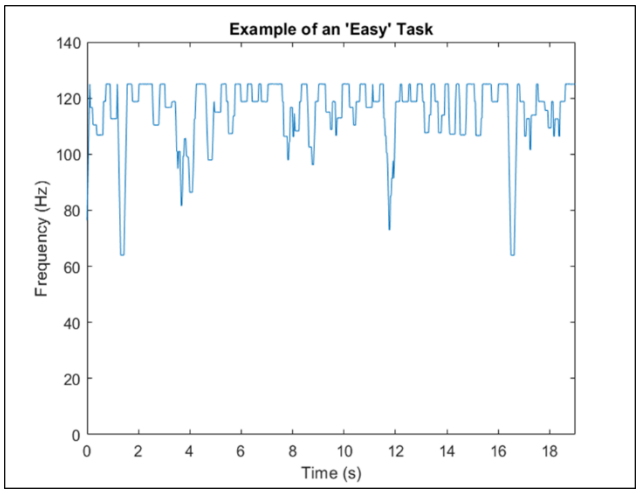

Figure 1 - Example of user input frequency for an "easy" task

To preprocess the difficulty scores, the raw TLX scores needed to be parsed from a series of *.csv filed. After processing the data, a histogram was created to determine how to quantify "easy" and "hard." To simplify label classification, a log transform [4] was applied to the data as to normalize the label's distribution. Figure 2 depicts the difficulty score histograms before and after the log transform. Three potential label sets were generated and are

[1] The author of this paper requested its citation not be provided for this report this report is public. Faculty grading this report may request further information from the author.

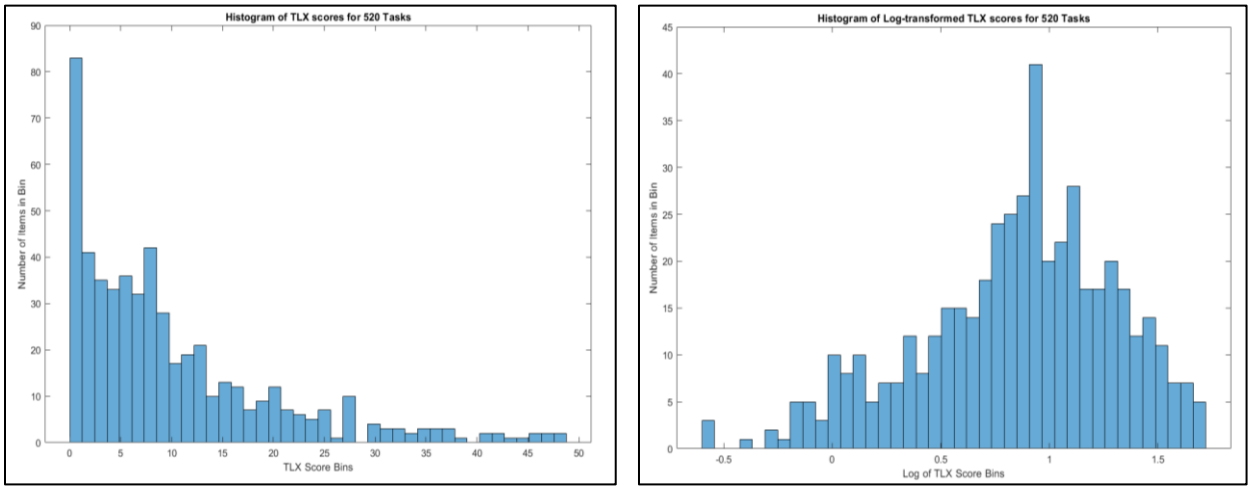

Figure 2 - Distribution of NASA-TLX scores before (left) and after (right) the log transformation

described in Table 1. Finally, a note: all preprocessing was done in MATLAB as to take advantage of the robotics toolbox for unpackaging *.bag files.

Label Set Name	Description
5050 Split	The TLX scores were divided into two equal parts, about the 50th percentile. This serves to test the feasibility of basic classification.
Top38	Based on the results in [1], the top 38 TLX scores were deemed difficult where as the other scores were classified as easy. [1] stipulates that only 38 scores might be 'difficult,' and proposes this classification task as future work.
EMH	The TLX scores were divided into three bins: easy, medium, and hard, divided about the 33rd and 67th percentiles. This serves to test the feasibility of more granular classification.

Table 1 – NASA TLX Score distribution discussion

Methods

Discussion with collaborators and faculty led to a two-method approach to classifying the dataset. The

Method One: Extract statistical and otherwise-generated features from the joystick inputs and test a variety of training using 10-fold cross-validation. Table 2 presents a list of extracted features and their significance. Table 3 presents a list of tested algorithms; the relationship of these features to their difficulty labels is an area of active research. Therefore, some of the algorithms presented in table 3 were via "educated guessing." Weka was used to train the models and report their efficacy. All three label sets were used for method one. Only the primary dataset was used for training and validation. Unless otherwise specified in Table 3, the algorithms were tested using default settings.

Feature Name	Feature Description
Standard Deviation	Standard deviation of the trial
Mean	Mean user input frequency
Median	Median user input frequency
Number of Outlier Features	Number of points less than 10% of the mean frequency; lower user input frequencies are thought to indicate user struggle
Standard Deviation of Outliers	Standard deviation of the aforementioned outlier set
Mean of Outliers	Mean of the aforementioned outlier set
Median of Outliers	Median of the outlier set
25^{th} percentile of Outliers	25^{th} percentile value of the outlier dataset; this and two other percentiles were investigated to see the impact of the distribution of the outlier set.
50^{th} percentile of Outliers	50^{th} percentile value of the outlier set
75^{th} percentile of Outliers	75^{th} percentile value of the outlier set
Lowest (most extreme) outlier	Most extreme value in the outlier set
Length of Dataset	Size of the dataset; this feature is strongly correlated with task difficulty [1], however, length of task is only known at the completion of a task.

Table 2 – List of extracted features

Algorithm	Discussion
ZeroR	For baseline classification
Naïve Bayes	To test classification based on features' conditional dependence
Logistical Regression	To test classification based on features' conditional dependence and consider features' correlation, if any
Decision Trees (J48)	To test if certain features allow for binary splits resulting in effective classification
Neural Network (Multilayer Perceptrons)	To test classification based on not-easily-observable relationships between features; tested with hidden network settings "a", 11, and 50
Nearest Neighbor (Ibk)	To test classification based on feature clustering; tested with NN=1 and NN=10

Table 3 – List of tested algorithms with brief description

Method Two: Feed the data into a recurrent neural network (RNN), specifically one consisting of long-short term memory units (LSTMs) to classify the data. The validity of this approach for this type of data is presented in [5]. This method was presented because of it's potential for future real-time implementation.

A generic LSTM was trained using Keras [6]. Keras is a high-level Python API for implementing neural networks; the API runs on top of PyTorch and simplifies the prototyping process for neural networks. The default parameters in the Keras LSTM package were used and only the data classified by the LSTM and the number of LSTM units were varied. Due to the 'proof-of-concept' nature of this project, the number of parameters varied were limited. Such was done at the advising of research collaborators and faculty. All three label sets were used in method two and both the primary and secondary datasets were used. The LSTMs were trained on 90% of the data and tested on 10% of the data.

Results and Discussion

Method One Results:

For all tests in the first part of this project, the Top38 label set was not able to build a classifier; using all of the methods listed in Table 3, none were able to beat ZeroR. This occurred largely in part because only ~7% of the examples were labeled as "difficult." This set was tested solely because of these points being explicitly mentioned in [1]. Since these negative results are very uninformative, they will not be discussed further. The appendix includes the raw results for this label set.

Figure 3 presents the classifier accuracy results across all features. It was observed that the length of the dataset strongly impacted classification accuracy; in other words, the longer a user required to complete a task, the more difficult the perceived task. The most accurate classifiers for both the 5050 Split and EMH datasets were the logistical regression and the neural network with 50 hidden layers. Figure 4 presents the performance of the best classifiers and their performance compared to ZeroR. While classification accuracy didn't surpass 90%, the best classifiers did improve from the baseline by 20%. The EMH label set experienced near identical behaviors, including identical preferred algorithms, as compared to the baseline metric; EMH results are also presented in Figures 3 and 4. Based on these results, it's hypothesized that 1) the features are not independent but rather are correlated and 2) given a large enough dataset, classifier performance would improve significantly.

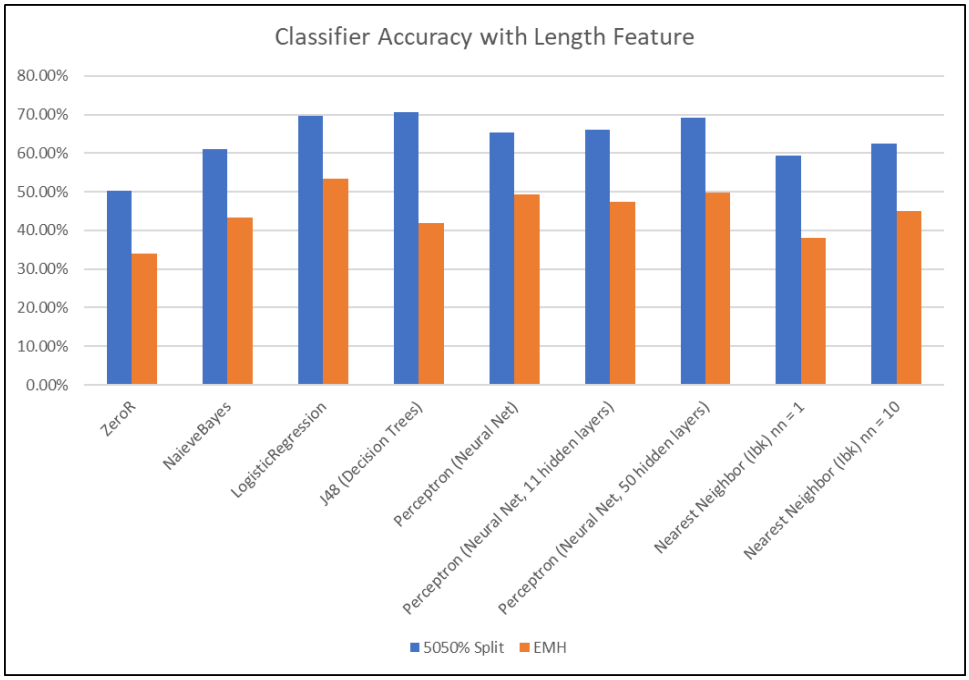

Figure 3 - Weka Classifier Results for both 5050% Split and EMH label sets for the once-filtered dataset for multiple classifier algorithms and their variations

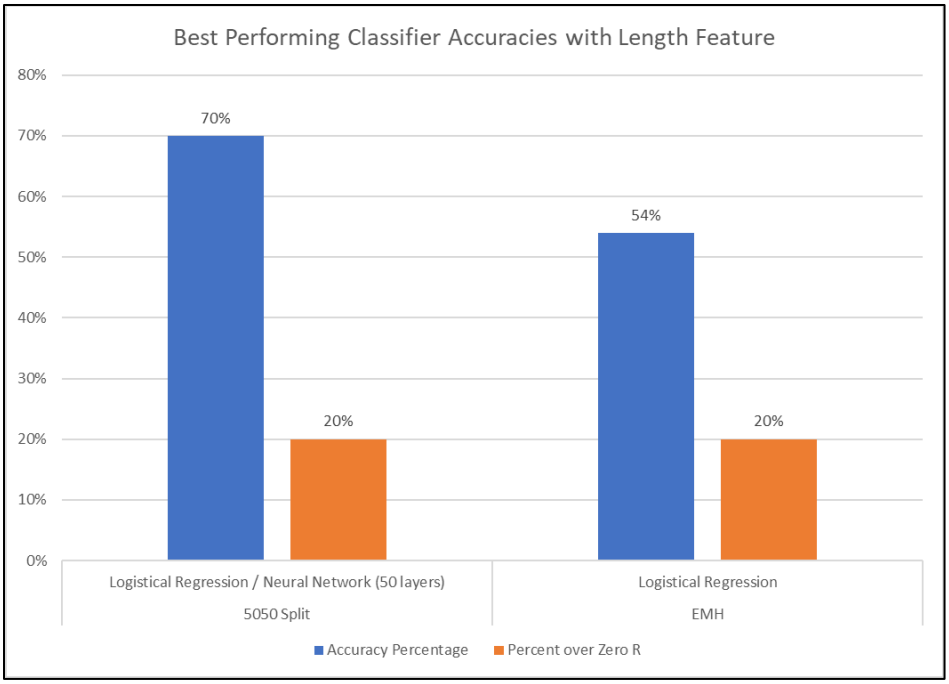

Figure 4 - Best performing classifier accuracy and accuracy vs. ZeroR performance for both 5050 Split and EMH label sets

Since task length is only known after task completion, the models presented in table 3 were trained again without the task length feature as to provide possible online performance insights. Figure 5 presents the classification performance of all algorithms tested and Figure 6 presents the performance of the best classifiers and their performance compared to ZeroR.

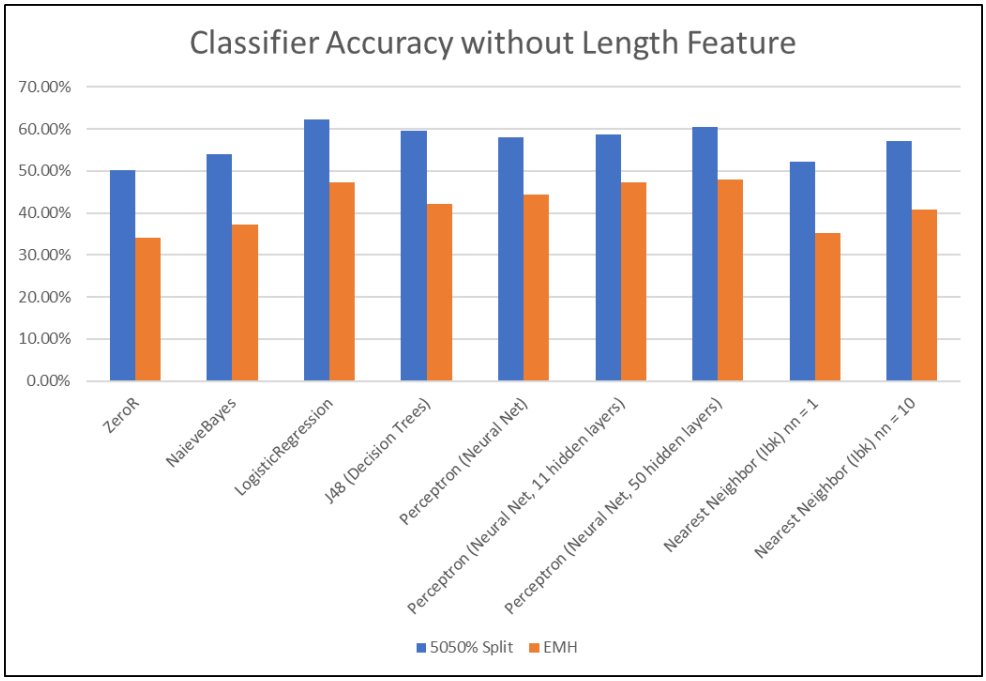

Figure 5 - Weka Classifier Results for both 5050% Split and EMH label sets for the once-filtered dataset for multiple classifier algorithms and their variations neglecting the length feature

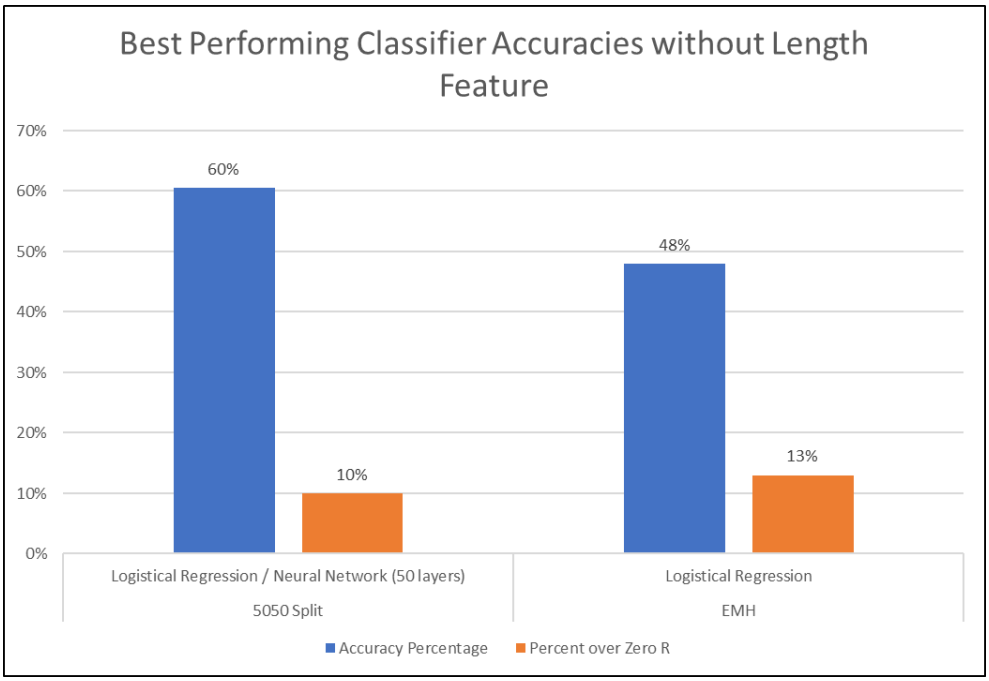

Figure 6 - Best performing classifier accuracy and accuracy vs. ZeroR performance for both 5050 Split and EMH label sets, neglecting the length feature

The peak classification performance fell from ~70% accuracy to ~60% accuracy with the removal of the length feature. The best classifiers were, again, logistical regression and the neural network with 50 hidden layers. The most important features appeared to be number of outliers and features describing the distribution of the outlier set. Yet again, the EMH label set experienced similar behaviors when compared to the baseline metric when not considering length as a feature. Figures 5 and 6 also present EMH results. The 10% improvement over the baseline indicates these features may still be useful for difficulty classification. A model to perform online classification likely requires a significantly larger dataset for usable accuracies; the use of user input frequency use alone for difficulty classification is not yet known.

Method Two Results:

As with method one, the Top38 label set was not very useful for classification. While investigated, results from this label set are only reported in the appendix.

Figures 7 and 8 present the classification accuracies for LSTM networks trained on both once- and twice-filtered data using the 5050 split and EMH label sets, respectively. Both figures compare models consisting of 10, 50, 100, and 200 LSTM memory units.

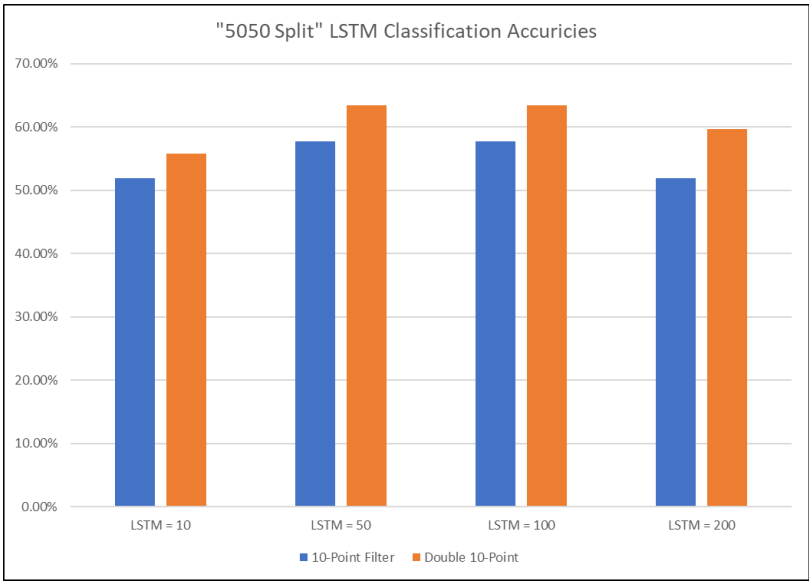

Figure 7 – LSTM model classification performance for the 5050 Split label set on both the once- and twice-filtered datasets

For the 5050 split data set, it was shown that an LSTM network of size 50 to 100 memory units performed best. The twice-filtered data yielded the best classification results yielding ~63% classification accuracy. This indicates that smoothing this dataset may help to accentuate extrema features that aide in difficulty classification.

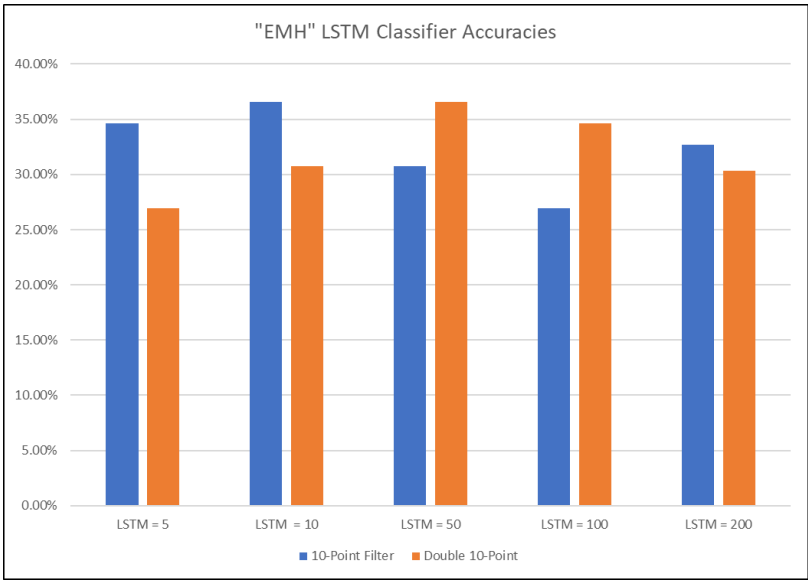

Figure 8 - LSTM model classification performance for the EMH label set on both the once- and twice-filtered datasets

For the EMH data set, it was shown that 10-50 memory units yielded the best performance. However, this might be noise as classification performance did not exceed 36%; this performance is close enough to ZeroR to state that the LSTM network was unable to classify the EHM label set.

A more advanced smoothing method, such as an exponential moving average may better accentuate features in this dataset for more accurate LSTM model generation. A list of methods to accentuate key features is a topic of future work. A larger dataset is likely necessary for proper model generation.

Conclusions and Future Work

Method One: Using both statistical and user-generated features, the highest classification accuracy discovered was just above 60% using either logistical regression or a neural network when not considering the length of the data set.

Method Two: the best performance observed using an LSTM occurred when using the twice-filtered data and 50 to 100 LSTM units. The performance was just above 60%, similar to the performance observed in method one. More advanced smoothing/filtering methods may improve classification accuracy.

The results from both method one and method lead to three conclusions:

1) Either this is the incorrect dataset for classifying task difficulty due to either its size and/or the tasks' design OR
2) User input frequency alone is likely not sufficient for task difficulty classification
3) Input frequency is somewhat likely to be a viable feature in a larger feature set for difficulty classification

Based on these findings, this project was a success. User input frequency may not be a good stand-alone metric for difficulty classification. However, this project will inform the design of a study that will review user input frequency as one of many metrics for eventual online, dynamic task difficulty classification and subsequent autonomy allocation. Such a study will consist of more tasks that are extremely different in difficulty levels; the details of the design remain an area of active research for the author for the author's PhD.

Acknowledgements

While all work in this project remain that of the author, the author would like to acknowledge his research colleagues of the argallab, in alphabetical order: Alex Broad, Mahdieh Nejati Javaremi, and Michael Young for their feedback, literature suggestions, data processing suggestions, and sharing of in-review papers to inform this project. Further, the author would like to acknowledge his friend from undergraduate study, Ravender Virk, of Amazon Web Services for providing an external perspective and always-appreciated comedic relief.

Appendix

To view the appendix, please visit:
https://github.com/chrisxmiller/FrequencytoDifficulty/blob/master/Appendix.pdf

Works Cited

Note: [1] is an unpublished and in-review works with IROS and cannot be shared at this time. Finally, up-and-coming papers of which the proposer is not a co-author are not shared for IP purposes.

[1] (In-Review) M. Young, C. Miller, Y. Bi, W. Chen, B. Argall "The Role of Kinematic Task Features in Robotic Arm Manipulation with Implications for Dynamic Autonomy Allocation." Submitted to International Conference on Intelligent Robots (IROS), 2018.

[2] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," Advances in Psychology, vol. 52, pp. 139–183, 1988.

[3] S. G. Hart, "NASA-task load index (NASA-TLX); 20 years later," Proc. of the Human Factors and Ergonomics Society Annual Meeting, vol. 50, no. 9, pp. 904–908, 2006.

[4] "Log Transformations." Online Statistics Education: An Interactive Multimedia Course of Study, Rice University, onlinestatbook.com/2/transformations/log.html.

[5] M. Wollmer et al., "Online Driver Distraction Detection Using Long Short-Term Memory," in IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 2, pp. 574-582, June 2011.

[6] Chollet, François. "Keras: The Python Deep Learning Library." Keras Documentation, Google, 2018, keras.io/.