A Study of Supervised Clustering Methods for Optical Mouse Trajectory Data from Tap Strap 2

1st Luke Heinrichs
Computer Science Department
Augustana College
Rock Island, Illinois, USA
lukeheinrichs21@augustana.edu

2nd Motti Kelbessa
Computer Science Department
Augustana College
Rock Island, Illinois, USA
mottikelbessa21@augustana.edu

3rd Ryan Freas
Computer Science Department
Augustana College
Rock Island, Illinois, USA
ryanfreas19@augustana.edu

***Abstract*—Abstract: This paper presents a study on the use of the Tap Strap 2 input device for creative applications. The Tap Strap 2 is a wearable device that allows users to input commands through a combination of finger taps. While the device has been marketed as a versatile input solution for a wide range of devices, we argue that it is not well-suited for everyday use, and that its main target audience is hobbyists and creatives. Our paper has two main goals: (1) to devise unique functionality for the Tap Strap 2 beyond that which is advertised by the manufacturer, and (2) to perform an analysis of the learning curve associated with learning the tap combinations of the Tap Strap 2. To achieve the first goal, we train a trajectory clustering machine learning (ML) model that can recognize gestures that resemble dial rotation, using trajectory data from the optical mouse sensor of the Tap Strap 2. Overall, the paper aims to demonstrate how the Tap Strap 2 can be used to expand the creative/artistic repertoire of hobbyists and creatives without the need for expensive specialized devices.**

***Index Terms*—trajectory analysis, Tap Strap 2, SVM, CNN**

I. Introduction

Computer Science is a very diverse field of study with many areas to explore. One of the fastest growing areas of this industry is human-computer interaction, or HCI. This area of computer science essentially focuses on the design of computer technology and how we can interact with these devices. Examples of this technology include the QWERTY keyboard, the smartwatch, and Virtual Reality machines [1]. The focus of our paper will be aimed at a family of devices that combine the functions of a computer mouse, a device that controls a cursor in a GUI, as well as a keyboard, an input device used to enter characters into a computer system. These devices are colloquially known as 'all in one devices.' Some examples, among many, of such devices are the Twiddler, the TapXR Bracelet, the Aula Excalibur One-Handed Mechanical Gaming Keyboard and Razer Tartarus v2 Gaming Keypad, One of these devices in particular, the Tap Strap, will be the focus of this paper.

Before we had the privilege of combining both the mouse and the keyboard into one device, they existed separately during their conception in the early 1960s. These devices stayed relatively the same through their different iterations up until the early 1990s when the company HandyKey Corporation quietly released the first version of an all-in-one mouse and keyboard combination. Their device, called the Twiddler, was marketed as a next generation input device that allowed someone to type with a single hand. The next huge innovation came in the mid-2000's during the dawn of Bluetooth technology. In 2003, apple released both the first wireless keyboard and wireless mouse. This represented a huge milestone in the tech industry, as this level of convenience was an extremely valuable commodity that every company wanted to obtain [2].

Subsequently, in 2019, a startup company called Tap Systems Inc. attempted to address the vast market opportunity of gadgets for virtual reality and smart wearables. Tap Strap 1 is a device that allows users to type characters by registering finger tap combinations by incorporating five different accelerometers and a gyroscope on a string of Thermoplastic Polyurethane (TPU) rings. The finger tap combinations are then processed by a computer chip located in the thumb ring of the Tap Strap, which then outputs the resulting letter or text as output. In addition, the thumb finger of the Tap Strap also features a 500 DPI (dots per inch) sensor, allowing the device to also be used as a mouse. The combination of these two functionalities allows for a wide range of use cases, including language independence, ambidextrous use, key mapping customization, and multi-platform interaction [3].

Despite its capabilities and the praise it received as an interaction solution that empowers people with visual impairment or limited mobility, Tap Strap 1 received significant feedback from customers and stakeholders that the device needed significant refinement before it could be considered a competitor to the ubiquitous keyboard and mouse. Then came, Tap Strap 2; unlike its predecessor, it included Air Gestures for swipes and gesture based media controls, an optical sensor with higher sensitivity (1000 DPI), improved battery life (2 hour advertised improvement), more ergonomic redesign of the thumb rest, and improved software and responsiveness. In essence, this successor addressed all the technical concerns that the customer had with their previous generation device. However, despite these improvements, Tap Strap 2 was not a commercial success either. At initial launch in 2019, the Tap Strap 2 was priced at $200, $20 more than its predecessor [4]. The retail price even reached a high of $ 249 in the first 3 quarters of 2022 [5]. This launching price made the Tap Strap 2 a very expensive computer interaction device which

was priced higher than many premium keyboards and mice. Even with the technical improvements, Tap Strap 2 inherited its predecessor's steep learning curve, making it difficult for users to learn and use the tap combinations. The gestures, i.e. tap combinations, were forced onto the user, the tap reads weren't as accurate as the ubiquitous keyboard and mouse, and, most of all, it was uncomfortable to wear and remove.

Just as Matt Jancer outlines in his article, *Review: Tap Strap 2 Input Device*, there is an argument to be made that the Tap Strap 2 is a wearable not meant for the day-to-day work life, but is rather ideal for situations when you are interacting with devices that don't have traditional peripherals such as gaming consoles, VR headsets, streaming devices and such. Yet, just like Jacer, we prefer using the default input system provided by the respective device. The Tap Strap 2 is just too cumbersome. As impressive a piece of technology as it is, for the non-hobbyist user, the Tap Strap 2 is "an answer in search of a question" [6]. And that is the audience that we'd like to specifically focus the efforts of our work on, the hobbyist and the prospective creative.

Our paper has two goals: (1) devise a unique functionality for the Tap Strap 2 beyond that which is advertised by Tap Inc., (2) perform an analysis of the learning curve associated with learning the tap combinations of Tap Strap 2 (§II-A3). With regard to our first goal, we are attempted to train a prototype for a trajectory mapping machine Learning (ML) model trained to recognize gestures that resemble dial rotation using data from the optical mouse sensor of the Tap Strap 2. The idea behind training such a model is to add additional functionality that the Tap Strap 2 could provide for creative tools or software such as Abelton Live, Adobe Premier, Adobe Photoshop, Blender, and/or Davinci Resolve. Specialized devices that already provide dial functionality to the aforementioned software are the Blackmagic Mini Panel, the Blackmagic Design Editor Keyboard, or the Tangent Ripple. If this prototype succeeds, our goal is to both to use predictions of our ML algorithm to use the Tap Strap as an alternative for the previously mentioned devices and demonstrate this via a simple GUI intended to mimic certain functionality within the aforementioned software. By doing this, we hope to provide a functionality for Tap Strap 2 that allows a prospective creative to expand their creative/artistic repertoire without necessarily having to buy expensive creative devices from the onset.

The paper is organized as follows. We start by describing prior work done in replacement/augmentative computer interaction technology and related work on trajectory analysis (§II). We then describe the data collected using Tap's Python SDK package[1], our methodology for choosing specific data points, and the preprocessing techniques we used (§IV). This is then followed by our experimental approach, in which we train prospective classification models and detail the development of a simple GUI we use to demonstrate the gesture functionality (§V). We then present the results of our approach along with a discussion about alternative computer interaction devices (§VI). Lastly, we make the limitations of our work clear and explore future improvements that can be done to improve our results (§VII).

[1]https://github.com/TapWithUs/tap-python-sdk

II. Related Work Technology

A. Wearables and All-in-ones

While Tap created a unique product with the Tap Strap 2, many other companies continue to expand in the HCI industry with a priority in efficiency and ergonomics. The use of a standard mouse and QWERTY keyboard continue to dominate computer and human interaction; however, some companies believe that the use of a different form factor may allow for a more comfortable and convenient working experience without a loss in work efficiency. In order to understand the potential of these devices, it is important to acknowledge the designs and features of some other wearable and all-in one devices.

1) Tap XR: Following the release of the Tap Strap 2, Tap has been developing its newest take on wearable technology called the Tap XR. Unlike its predecessor, this device is worn on the wrist, rather than around each finger to accomplish a design that is more comfortable to wear. Using a retractable sensor, it tracks the movements of each finger, allowing for similar functionality as the Tap Strap 2. However, due to its new design, the device only functions as a keyboard and controller. Mouse movements are now replaced with controller-based options, meaning the device will be incompatible with some applications. The Tap XR has a planned release in quarter one 2023 alongside open-source development kits available for additional implementation.

2) Twiddler 3: In 2014, the company Tek Gear released their adaptation of a mobile keyboard and mouse combined in a single unit. Their goal was to create a one-handed device capable of fitting in the palm of one's hand, allowing for convenience and portability. Their design resembles a Meta Quest 2 controller that fits within the palm of one's hand, allowing for easy storage. In order to accomplish the functionality of a full keyboard in a small form factor, twelve alphabetic keys are located on the side of the device, similar to a telephone keypad. Three function keys are located above these, allowing users to switch letters within a single button. For the mouse, an eight-directional joystick is in the top of the device, where one's thumb may be used while the other fingers are typing. With this layout, they claim users can achieve 30-60 wpm with some practice, with the potential to reach up to 260 wpm when customized to a user's needs using macros. The Twiddler 3 can be connected via Bluetooth 2.0 for a wireless connection, or through a wired connection using the USB connection on the bottom of the device. To this day, their documentation continues to be updated.

3) Azeron Cyro: In April 2023, Azeron, a company that specializes in gaming peripherals, will release the Azeron Cyro, an all-in-one mouse and keypad designed specifically for gaming. The device prioritizes comfort and ease of use, with a design that places the hand in a natural grip and an adjustable strap that holds the hand securely in place. The Cyro also features a joystick on the top of the device, which allows for

omnidirectional movement, as well as a 5-way switch that can be easily accessed with the thumb. Along the side, there are 16 blank keys, which are available to all other fingers. This allows for a wide range of customization options, enabling users to assign different commands and macros to the keys, thereby allowing both keyboard and mouse functionality at the same time. However, due to the lack of buttons, the Cyro is not intended for typing. Instead, it is designed for gaming. With this in mind, the device will connect to a computer USB for a faster connection.

While all these aim to replace a mouse and keyboard, their differences in design allow for different and specialized use cases. However, due to the complexity and customizable focus within these products, users are required to learn a new method of typing, decreasing initial efficiency. In a study conducted by Aalto University and the University of Cambridge, the average words per minute among normal users when typing was 52. However, those professionally trained averaged between 70 and 90 with a maximum of 120 [7]. When comparing the estimated typing speed given by Tap and Tek Gear, both the Tap Strap 2 and Twiddler fall under the average words per minute of experienced typists at between 30 and 60. However, with their Bluetooth connectivity and small size, these devices also advertise their use on mobile devices such as tablets and phones. According to a similar study, users typing on a mobile screen achieved an average of 36.17 words per minute, with a high of over 80 [8]. When comparing these average words per minute to the advertised speeds from Tap and Tek Gear, both devices claim that users will be able to type within this average as seen in our graph in Fig. 3. With this, we believe the Tap Strap 2 may successfully replace a keyboard and mouse for some users; however, using its other sensors, it is capable of more.

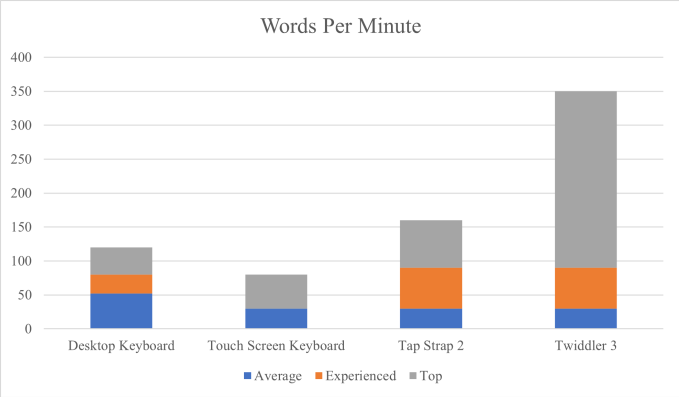

Fig. 1. Words per minute across devices

B. Related Work on Trajectory Mapping/Analysis

Before quoting research that looks at trajectory classification, a brief introduction on trajectory mapping/analysis might be necessary. Trajectory analysis is a method used to study the behavior of a system or an object over time. This can include analyzing the movement of a physical object, such as a vehicle or animal, or the progression of a system or process, such as the spread of a disease or the behavior of a stock market. The goal of trajectory analysis is to understand the underlying patterns and trends in the data, and to use this knowledge to make predictions or identify areas of concern. Machine learning techniques, such as time series analysis and predictive modeling, are often used to analyze trajectories and make predictions. Trajectory analysis is commonly used in fields such as robotics, transportation, and finance.

III. Available Development Tools

In order for the Tap Strap 2 to discern different taps and combinations, each finger is assigned an increasing integer value doubling for each finger starting at 1; the thumb is assigned 1, the index is assigned 2, the middle is assigned 4, the ring is assigned 8, and the pinky is assigned 16. Using these values, the Tap Strap is able to assign each combination of finger presses to a unique integer. However, each finger and combination allow for double and triple taps by comparing the current tap data with the previous within a small window of time. However, the maximum time in between each tap in which the device can register is 65535 milliseconds and cannot be adjusted beyond that [9]. These integer values and combinations can then be translated into letters as seen in the out of box Tap Strap, but using code, these integers can be reassigned for multiple uses. The device also utilizes an optical sensor on the thumb in order to track mouse movements. The sensor works on a two-dimensional coordinate plane, in which it reports the change in its x and y position relative to its previous position. Tap has released three official Software Development Kits (SDKs) for developers programming for Windows, IOS, and Android; a unity plugin is also available. However, because these SDKs are open source, a python SDK is also available for developers to use. Using this, the Tap Strap 2 can be used in four different modes: text mode (which allows for normal keyboard and mouse operation), controller mode (where all input is sent directly to the SDK), text and controller mode (where the Tap Strap 2 functions as a mouse and keyboard while also sending data to the SDK), and raw data mode (which sends raw sensor data to the SDK) (§1).

IV. Sensor Data Collection

As mentioned prior, we are using Tap's Python SDK to extract raw sensor data intended for the training and evaluation of our trajectory mapping model. As mentioned in the previous section, this SDK allows us to send and receive data and events by establishing a Bluetooth Low Energy (BLE) connection with the Tap Strap 2. This allows us to get raw data from the sensors embedded within the Tap Strap 2. As noted by Mrazek et al. in their work on the input accuracy of Tap Strap 2, Tap's Python SDK doesn't allow us to obtain sensor data and interpreted input simultaneously [10]. However, this isn't much of an issue to our work, since the circular gestures we are intending to get data for are novel to Tap Strap 2's internal firmware. This makes the firmware based interpretation of our

gestures unimportant in labeling the optical mouse sensor data. Therefore, we labelled all the data points manually.

We collected data in the following manner. First, we wrote a Python script, using the Tap's Python SDK with the input mode set to "controller and text" (where the Tap Strap 2 functions as a mouse and keyboard while also sending data to the SDK) to dump mouse movement data, i.e. relative (x, y) coordinates into a CSV file. Since the gestures we would like to train our trajectory analysis/mapping model to recognize mimic the rotation of a dial, we recorded circular relative (x, y) coordinates by tracing curved geometric shapes that resemble a circle such as an ellipse, a cardioid, circles, ovals, and ellipses. To record such data points, with the script running in VS Code (version 1.72.2) and the Tap Strap 2 worn on the right hand, we traced the aforementioned circular paths of different sizes in a Word document: picture show in Fig.2. We traced these circular paths both in clockwise and counter-clockwise direction because this resulted in a significant difference in the shape of the traced path, although paths drawn in either direction belonged to the same trajectory class-class circle. This was repeated 100 times in each direction. Although these trajectories were traced, the circular shapes produced were in no way a perfect trace of the path in the Word document.

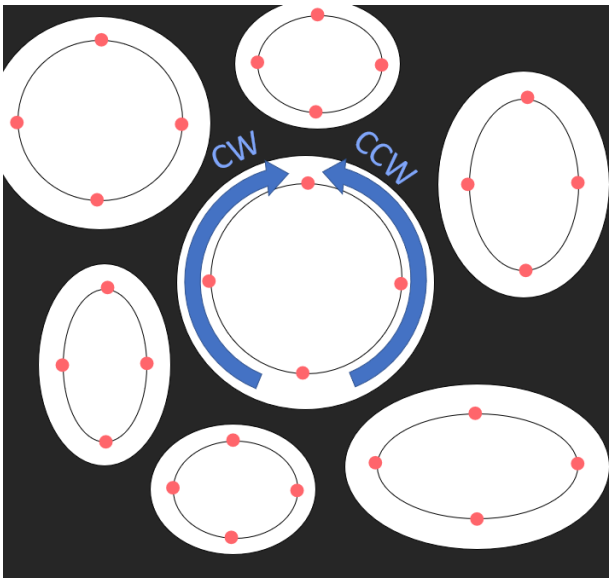

Fig. 2. Tracing paths used to record data mouse movement data

In addition, considering that random mouse movements such as side to side, up and down, or zig zag movements could result in false positives, we recorded 100 data points for random movements in both clockwise and counter-clockwise direction. In total, we recorded 300 pairs (meaning an x and y for each) of relative coordinates separated into three CSV files: one for circular paths traced in a clockwise direction, one for circular paths traced in a counter-clockwise direction, and one for random movements. Since these data points were acquired from an optical sensor, each sample contains the (x, y) coordinates of the moving object at every instant relative to the origin. This makes it difficult to handle for classification tasks and necessitates some pre-processing.

Mainly, pre-processing our data encompassed two main aspects: changing the trajectory coordinates from relative to non-relative, and then transforming the feature space or normalizing the (x, y) trajectory coordinates. The first of these involved setting (0, 0) as a starting point for each pair of trajectory data and adding each subsequent record of the relative coordinate pair (both x and y) to the one before it. The next aspect of our pre-processing techniques involved centering the both x and y coordinates around zero, which was done using this equation for Z-score:

$$z = \frac{(\mathbf{x} - \mu)}{\sigma} \tag{1}$$

where $\mathbf{x}$ is a 2D-array (2 columns) of x and y coordinates from our sample, μ is the mean of the sample, and σ is the standard deviation of the sample. Z-score normalization was implemented using the Scikit Learn library of Python [11].

V. Approach

Our experimental setup is composed of 2 distinct parts: (1) training a classification model to discern between relevant and irrelevant gestures, (2) a simple software intended to demonstrate the functionality we introduced in the introduction sectionc(§I).

A. Gesture Classification

1) Lazy Classifier: In addressing the first part of our approach, we decided to evaluate what type of classification model, among many, would be best suit our data set using Lazy Predict [2]. Lazy Predict is a Python library that is built on top of popular machine learning libraries such as scikit-learn, xgboost, and lightgbm. The library is designed to automate many of the tedious and repetitive tasks that are associated with machine learning, such as data preprocessing, feature selection, and model tuning. The library has a built-in function called "automl()" which can automatically select the best model for a given dataset. This function takes in the dataset as an input and returns the best model based on its performance on the dataset.

Our dataset was split up and fed into the model as follows. The whole data set was imported into the training notebook and then split using sklearn's *train_test_split* into *test_size*=0.2 and *random_state*=55. The parameters in *LazyClassifer* were left at default, besides the following: *verbose*=0, *ignore_warnings*=True, *custom_metric*=None.

2) Support Vector Machine (SVM Classification): Although lazy classifier covers most of the classification algorithms available, Support Vector Machine (SVN) classification is an important one that we couldn't overlook. Just like other classification algorithms, SVMs find the best boundary (or "hyperplane") that separates the different classes in a dataset by choosing a boundary that maximizes the margin between classes [12]. In our case, SVMs are particularly useful where

[2]https://github.com/shankarpandala/lazypredict

the data is not linearly separable, meaning that a straight line cannot be used to separate the classes. In these cases, SVMs use a technique called the kernel trick to transform the data into a higher-dimensional space, where it may become linearly separable [13] [14].

As in the previous case, Our dataset was split up and fed into the model as follows. The whole data set was imported into the training notebook and then split using sklearn's *train_test_split* into *test_size*=0.2 and *random_state*=55. We then used the *predict* method on the model and passed our split as a parameter for each training iteration for our model.

B. Implementing the Tap SDK

For our use case, we will be using controller mode, as it disables the normal text output of the device and sends all its information to the SDK. With the Tap Strap sending all of its data to the SDK, we are able to take the integer values assigned to each finger and associate them with functions and objects we have created. Each finger contains a name, an assigned function, and the tap code sent from the Tap Strap. Using this, users are able to assign a use case to each individual finger and label it with a corresponding name. When a finger is pressed down, our prototype will identify which finger to activate and give a vibration as feedback that the input has been registered. It will then send the corresponding function to a designated destination. While Tap has developed their own mapping tool in which users can remap their Tap Strap 2, creating our own allows for easier cooperation of data between both the tapping and optical sensor we will be using. To activate the device, the user must tap the index, middle, ring, and pinky finger at the same time; doing so will allow the functionality activation to be toggled on and off quickly as to avoid accidental presses while using a mouse and keyboard. When a finger is tapped, it will activate a feature and allow the optical sensor to send input, meaning the sensor will not interfere with one's work until it is specifically needed. Once activated, the user will be able to increase and decrease the values of the selected function's tool using gestures that resemble the rotation of a dial. When a user completes a full motion of a circle using the optical sensor, the prototype will increase or decrease the value of the currently selected setting. Rotating in a clockwise rotation results in an increase, and a counter-clockwise results in a decrease. The interval value is adjustable, allowing for the user to quickly manage values at a higher or lower rate of change. When the user is done, they can simply deactivate the device using the same four finger combination above. With these implementations, we believe the Tap Strap 2 may serve as a suitable device for increasing productivity while working alongside a traditional mouse and keyboard.

C. Tap Strap GUI

To showcase the new features we created for the device, we created a functional GUI using the python framework named tkinter[3]. This is the only framework that is built into the standard Python library. Although this framework may lack the modern sheen of alternative GUI packages, it prioritizes functionality and having a modest learning curve. We first worked on linking our Tap Strap device to our GUI to display inputs fed into it from the device. From there, we created four different text fields to display the Tap Strap inputs corresponding to the text fields present in common photo editing apps such as Adobe Photoshop: altering the hue saturation curve, applying color correction, adjusting shadows, and adjusting exposure. Each of these fields had it's corresponding label on top of it. We also include an additional text field to keep track of and modify the "Offset Value" by which the values in the text fields are updated. We also included a widget at the top of our GUI to display whether the Bluetooth connection between the Tap Strap 2 and the SDK was correctly established or not.

[3]https://docs.python.org/3/library/tkinter.html

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
KNeighborsClassifier	1.00	1.00	1.00	1.00	0.02
SVC	1.00	1.00	1.00	1.00	0.02
AdaBoostClassifier	0.97	0.95	0.95	0.97	0.13
XGBClassifier	0.97	0.95	0.95	0.97	0.05
RandomForestClassifier	0.97	0.95	0.95	0.97	0.32
NuSVC	0.97	0.95	0.95	0.97	0.06
BaggingClassifier	0.97	0.95	0.95	0.97	0.04
LGBMClassifier	0.97	0.95	0.95	0.97	0.05
DecisionTreeClassifier	0.97	0.95	0.95	0.97	0.02
ExtraTreesClassifier	0.97	0.95	0.95	0.97	0.14
LabelPropagation	0.93	0.93	0.92	0.93	0.02
LabelSpreading	0.93	0.93	0.92	0.93	0.02

Fig. 3. Results of Lazy Classifier

VI. Results and Discussion

VII. Limitations and Future Work

The limitation of our work could be classified into two parts: data dependence and fatigue caused by repetitive circular gestures.

A. Data Dependece

In general, trajectory analysis/classification can be approached from two different viewpoints. On one hand, there is a data-driven approach which aims to classify trajectories by grouping similar ones and identifying outliers, which is the focus of our project. On the other hand, the second viewpoint aims to categorize trajectories into predefined categories or classes [15]. Each of these have their own limitations, but

in our case the data dependence of the viewpoint we chose might be the biggest limitation. It may not generalize to other datasets, and a trend discovered in our data may not be what is desired for someone else. Furthermore, it is difficult to determine whether the clusters discovered in our work are meaningful or merely an artifact of the data we collected [16]. Therefore, although our results work well for our use case, there is no guarantee that our trajectory clustering model identifies the desired trends or patterns in any other data set or use case.

To improve the results of our study, we aspire to incorporate the following aspects in our future work, among many others:

1) Use a large and diverse dataset: Having more data can help to better identify patterns and trends in the data. Furthermore, using a diverse dataset that includes different types of trajectories can also help to improve the generalizability of the results in our study [17].
2) Use advanced machine learning techniques: Trajectory analysis is a complex task, and thus it may benefit from the use of advanced machine learning techniques such as deep learning, which can help to improve the accuracy of the results. The complexity of the techniques might however be data constrained.
3) Incorporate domain knowledge: Incorporating domain knowledge into the analysis can help to improve the interpretability of the results and to better understand the underlying mechanisms of the data. This might also include adding more classes when labelling our data.
4) Handling missing and noisy data: Handling missing and noisy data is a common problem in trajectory analysis. To improve the results, we propose the use of methods that are able to handle these types of data effectively [18].

Other ideas that we will consider incorporating are the following: incorporating temporal information, incorporating spatial information, evaluating the results with better metrics, and using ensemble methods.

B. Physical Fatigue

Although the gesture we trained our trajectory classification model on mimic that of a dial, its resulting output the gesture we trained for doesn't mimic the output behavior of a normal dial. On most electronic devices such as a lamp or a speaker, a dial is made of a variable resistor such as a potentiometer on which a full rotation of the shaft corresponds to a full range of resistance, going from minimum to maximum resistance. And even more specifically, for example, a potentiometer with a range of 0-100k ohms, when turned by 90 degrees or by quarter of a circle, will change the resistance by 90/360*100k = 25k ohms. As compared to our implementation of a gesture based dial, this seems much more efficient and would cause less fatigue.

Therefore, for future iterations of our work, we would like to evaluate the prospect of training a trajectory clustering model that recognizes paths that resemble the arc of a circle instead of a fully circular path. This might reduce the physical fatigue caused by making multiple circular movements, while also providing finer control to the user.

References

[1] N. Roussel, "Looking back: a very brief history of hci," *Unpublished working draft*, pp. 1–2, 2014.

[2] P. Suresh, J. V. Daniel, V. Parthasarathy, and R. Aswathy, "A state of the art review on the internet of things (iot) history, technology and fields of deployment," in *2014 International conference on science engineering and management research (ICSEMR)*, pp. 1–8, IEEE, 2014.

[3] "Tap strap 1 keyboard wearable," Jan 2023.

[4] B. Levin, "The tap strap 2 turns your hand into a keyboard and mouse | cnn underscored," Dec 2019.

[5] K. Keepa, "Amazon price tracker," 2021.

[6] M. Jancer, "Review: Tap strap 2 input device," May 2020.

[7] V. Dhakal, A. M. Feit, P. O. Kristensson, and A. Oulasvirta, "Observations on typing from 136 million keystrokes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, (New York, NY, USA), p. 1–12, Association for Computing Machinery, 2018.

[8] K. Palin, A. M. Feit, S. Kim, P. O. Kristensson, and A. Oulasvirta, "How do people type on mobile devices? observations from a study with 37,000 volunteers," in *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[9] T. S. Incorporated, "Tap ble api documentation," 2019.

[10] K. Mrazek and T. Khan Mohd, *Using LSTM Models on Accelerometer Data to Improve Accuracy of Tap Strap 2 Wearable Keyboard*, pp. 27–38. 01 2022.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] S. Yue, P. Li, and P. Hao, "Svm classification:its contents and challenges," *Applied Mathematics*, vol. 18, pp. 332–342, 09 2003.

[13] D. Wilimitis, "The kernel trick," Feb 2019.

[14] M. Hofmann, "Support vector machines-kernels and the kernel trick," *Notes*, vol. 26, no. 3, pp. 1–16, 2006.

[15] A. Teatini, "Movement trajectory classification using supervised machine learning," 2019.

[16] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," *CoRR*, vol. abs/1802.06971, 2018.

[17] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[18] A. Zonta, "Handling noise and missing values in sensory data," Jun 2019.