Can we write on tablets with a conventional pen? Explore hand imprint dynamics for handwriting recognition

Author:
Thomas Berkane

Supervisor(s):
Chenyang Wang, Lucas Burget

Professor:
Pierre Dillenbourg

January 6, 2023

Contents

1 Introduction

People today commonly use digital tablets to take notes by writing, which usually requires a special stylus. The technology that makes this possible relies on the tablet detecting the location of the pen tip as it touches the screen, which is then rendered as part of the handwriting. However, an important aspect of handwriting that is often overlooked is the hand imprint on the screen when the hand is resting naturally on it. Indeed, most modern digital tablets have the ability to detect the full palm imprint using mutual-capacitance sensing technology.

The static and dynamic features of the hand imprint during the handwriting process most likely contain valuable information that could improve handwriting recognition when using a conventional pen on a tablet. This is what this project seeks to investigate.

Section 2 presents the relevant literature which is used as a starting point for this project.

In Section 3, the material and methods used are described. To begin with, I explore different alternatives for devices and ways to access their capacitive data. Then, I show how the different hardware sensors and software are used to collect the data from participants. Finally, I explain the three different methods I have attempted to do handwriting recognition on this data.

In Section 4, these three methods are evaluated and we gain some insight into how they behave and how they might be improved.

In Section 5, I conclude and discuss further steps which could be taken to continue advancing the project.

2 Literature Review

The literature does not contain any information about using capacitive data for handwriting recognition. However, capacitive screens have been used in several instances for hand pose estimation.

In [4], a custom capacitive surface is built and is used alongside a depth camera and a hand tracking camera to collect data. With this data, a model is built which makes predictions for hand pose based on the capacitive data. Reproducing this model could thus be a good intermediate step for the objective of handwriting recognition.

[5] also uses capacitive data for 3D hand pose estimation. However, it uses a Samsung tablet instead of a custom capacitive surface, making it more relevant to our project. Moreover, the paper describes preprocessing steps such as normalization. The authors also provide the code for the Android app which runs on their tablet, which can be useful as a basis for our own data collection app.

3 Material and Methods

3.1 Accessing Raw Capacitive Data on a Commodity Device

Most modern smartphones and tablets use touchscreens that rely on mutual capacitance as their primary means of input and output. While mobile operating systems like Android only provide access to 2D touch coordinates, it is sometimes possible to modify the OS to obtain the raw measurements of capacitive touchscreens, also known as capacitive images. These raw measurements are obtained from the touch controller, which is a hardware component that manages the touch input for a device.

To modify the OS of a tablet, it is necessary to root the device and install a custom kernel. This process can be complex and the steps can vary significantly from device to device, so I took time to research device options.

The first device considered was the LG Nexus 5 smartphone, as there already exists a compiled modified kernel, detailed instructions on how to install it, and a convenient Android library to access the capacitive data[8]. However, this option was set aside as the device would be too small to write on compared to a tablet.

The second option was the Samsung Galaxy Tab S6 tablet. As a modified kernel for it is not yet available, it would be necessary to modify and build the code for an open source kernel ourselves. In particular, I decided to go with the LineageOS kernel[13] as it is quite popular and well documented, adapting the steps given by the RainCheck project[14] to modify the Nexus 5. I spent some time investigating this possibility and arrived at the conclusion that the main modifications would have to be made in the synaptics_i2c_rmi.c file. In the end, this option was not retained.

Finally, I decided to use the Samsung Galaxy Tab S2. This is an older tablet (2015), meaning that it is incompatible with more recent pens such as the Samsung S Pen. However, I was able to obtain a modified kernel for it, along with instructions to install it which were kindly provided by Prof. Sven Mayer. Furthermore, he developed an Android app[9] to record capacitive images which I was able to use as a basis for my own data collection app.

3.2 Data Collection

3.2.1 Sensors

A possible intermediate step for the handwriting recognition would be to first predict the hand pose from the capacitive data, as in TouchPose[4]. To reproduce the model in this paper, we need ground truth data for the hand's pose and for depth while writing.

To record hand pose data, I use the Leap Motion[12] along with a script which uses its Python SDK to save the recorded data to JSON. A simple script has also been written for visualizing the recorded positions of the hand's joints. I also considered using Google MediaPipe, which has the advantages of being open source and doesn't necessarily have to run in real-time compared to Leap Motion. However, it is reported to be less accurate than Leap Motion.

To record the depth data, I use a RealSense[10] depth camera which is mounted above the tablet. Both the depth image and RGB image are recorded using the RealSense Viewer software. Another option could have been to use a Kinect camera, which also has a library for detecting the finger tip positions.

The data from these two sensors, together with the capacitive data, are matched into triplets by finding the nearest timestamps.

The complete setup for writing on the tablet with the two sensors is shown in Fig.1. This setup is appropriate for a right-handed person. For a left-handed person, it would have to be mirrored to ensure that the Leap Motion's field of view is aligned with the user's palm. The RealSense is not shown as it placed directly above the center of the tablet, at a distance of about 60cm. Also note that headphones with buttons for volume control are used to be able to navigate to the next sample more conveniently.

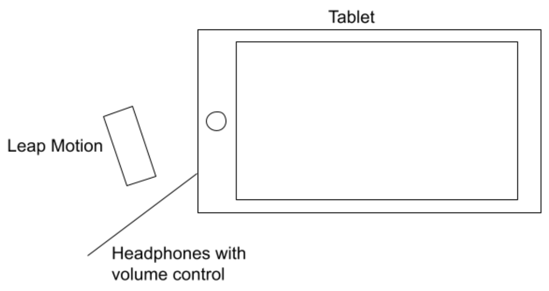

Figure 1: Setup for recording the capacitive data from a right-handed user

3.2.2 Stylus

To train the handwriting recognition model, it would be useful to have access to other data in addition to the capacitive images, such as position of the pen tip, pen pose or pen pressure. To this end, during the data collection we have participants write with a stylus. Three different styluses were considered:

- **Samsung S Pen**: this stylus has an SDK[11] with features such as gesture recognition. However it is only compatible with the Samsung Galaxy Tab S6.

- **Sonar Pen**: it has a palm rejection feature, but I ended up implementing this myself based on the capacitive data.
- **Conventional stylus with rubber tip**: it does nothing else than write on the screen. This is the option I chose for its compatibility.

3.2.3 Android app

An Android app has been developed which records capacitive data while a participant writes a sequence of characters on the tablet's screen. The app is designed to run while the hand pose and depth data are being recorded simultaneously. Also note that to function properly, the app must run on a tablet whose kernel has been modified as described above.

The app's starting screen allows to choose several settings:

- **Character sizes** to include. Large, Medium and Small characters can be used. This option was included to study the effect of writing size on the handwriting recognition.
- **Number of samples per character**: how many samples to ask the user to draw for each character. This was included to generate more data. A typical value is 5.
- **Writing style**: Guided and/or Freehand. Guided means that the user will have an outline showing the size and shape of the character to draw, to generate more consistent samples. Freehand means that only a box will be shown, to indicate the approximate size the character should have. The latter option will produce more diverse characters.

It is also possible to choose between right- and left-handed modes (which will require flipping the tablet around). The participant's ID can also be selected for syncing with the hand pose and depth data. Finally the starting screen allows to select between two different modes: capacitance testing, and data collection.

The capacitance testing mode is used to make sure that the capacitance screen is working properly and to visualize the generated hand imprints in real-time.

The data collection mode presents the participants with a sequence of characters (depending on which settings were selected) to draw using the stylus. The volume down button is used to navigate to the next character. The volume up button is used to wipe the recorded capacitive data for the current character in case a mistake is made.

Concerning writing on the tablet's screen, two things are noteworthy. The first is that it is only possible to draw the character if the user's palm is touching the screen. This is to ensure that the palm's capacitive image is being recorded at all times. The second is that since palm rejection is provided neither by the tablet nor by the stylus, I had to implement it based on the capacitive data. My implementation finds the connected components in the capacitive image. It then only retains the connected components corresponding to the stylus tip, which is found by analyzing each connected components' area and x-coordinate. This algorithm is run for each capacitive frame in order to detect in real-time which touch on the screen corresponds to the pen.

After a data collection session is over, we are left with a JSON file containing mappings from timestamps to capacitive images, which are arrays of dimension 37*49, with values in the range [-30,230].

For now, the capacitive data has been collected for three participants: my two supervisors and myself.

The code for the data collection app and scripts can be found at [7].

3.3 Handwriting recognition

I attempt three methods for recognizing which character is being written. The character prediction is performed on the sequence of preprocessed capacitive images corresponding to that character. Each sequence of preprocessed capacitive images has a mean length of 28 images with a standard deviation of 7.

The code for the handwriting recognition can be found at [6].

3.3.1 Preprocessing

In order to clean the raw capacitive images, the following steps are taken:

- Pixels with negative values (which are caused by parasitic capacitance) are set to 0.
- The imprint corresponding to the stylus tip is detected and the corresponding pixels are set to 0, to align with the objective of the project of writing with a normal pen (which wouldn't generate a capacitive trace).
- Blank capacitive images (such as those found in between characters) are removed.

3.3.2 Prediction with centers of mass image similarity

In this method, the sequence of capacitive images is first transformed into a sequence of centers of mass in an attempt to track the movement of the palm during the writing. This sequence of centers of mass is then combined into a single capacitive image, which thus represents through which pixels the center of mass of the palm went. Finally, a bounding box for the pixels is found and the capacitive image is cropped accordingly to account for differences in absolute palm position.

For example, the resulting image for the digit 0 is shown in Fig.2.

To obtain a prediction, the computed image is then compared to the computed image of a reference sample for each of the possible characters. The predicted character is then the character whose computed image has the highest similarity with that of the sample being predicted. Several similarity metrics were compared and the one giving the best performance was Spectral Angle Mapper (SAM)[2].

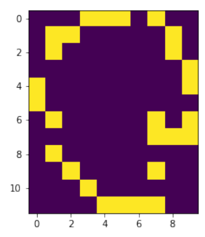

Figure 2: Center of mass pixels for a large freehand 0

3.3.3 Prediction with dynamic time warping on trajectory

In this method, the same sequence of centers of mass is extracted as in the previous method. This sequence is then transformed by taking the difference between two subsequent centers of mass to obtain a sequence 2D vectors, representing the trajectory of the palm's movement. These vectors are then normalized.

For example, the resulting vectors for the digit 0 is shown in Fig.3.

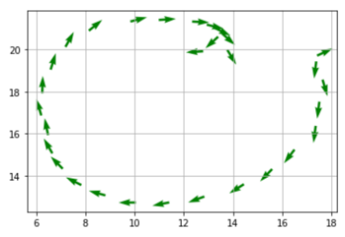

Figure 3: Trajectory for a large freehand 0

To obtain a prediction, the procedure is similar to the previous method. The difference is that the similarity is obtained by taking the inverse of the value computed by Dynamic Time Warping (DTW)[3], an algorithm to compute distances between sequences of different lengths.

3.3.4 Prediction with optical flow

In this method, the Optical Flow algorithm is used to extract trajectories from the sequences of capacitive images. The algorithm is tuned by only keeping trajectories of length at least 7, and by imposing a conservative maximum distance between two subsequent points in the trajectory, resulting in approximately 10 trajectories per sequence of capacitive images. These extracted trajectories are then converted to sequences of 2D vectors as described in the previous method.

An example of extracted trajectories for the digit 1 is shown in Fig.4.

To obtain a prediction, the set of extracted trajectories is then compared to the set of extracted trajectories of a reference sample for each of the possible characters. To compute the similarity between these two sets of trajectories, we first compute the similarity (using DTW) for each pair of trajectories between those extracted for the reference character

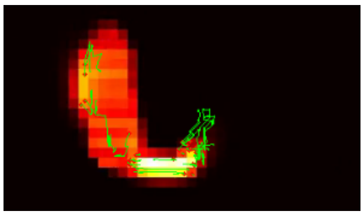

Figure 4: Trajectories extracted by optical flow for a large freehand 1

	Guided	Freehand
Large	0.025	0.025
Medium	**0.15**	0.125
Small	0.1	0.1

Table 1: Accuracy with centers of mass image similarity method for the different combinations of sizes and writing styles.

and those extracted for the character being predicted. We then match up the trajectories found in each of the two sequences of capacitive images by choosing the pairs with highest DTW similarity. The final similarity is then computed by taking the mean over the DTW similarities of the chosen pairs.

4 Results

The three methods presented are evaluated with the accuracy of the character prediction. In particular, I focus on recognizing the ten digits. Thus, this is a classification task with 10 classes.

For each combination of digit, size and writing style, 5 samples have been recorded from a single subject, with one of these being used as a reference for the prediction. To avoid bias in the choice of the reference digit among the 5 samples, each of them is chosen turn by turn, with the 4 remaining digits being used for evaluation. The reported accuracy is then the mean of the 5 accuracies obtained.

Instead of using reference digits which are always of the same size and style as the digit being predicted, I have also tried always using a large guided digit, which led to an approximate decrease of 15% in the mean accuracies.

4.1 Results for centers of mass image similarity method

The accuracies for the different combinations of sizes and writing styles are shown in Table 1. For each combination of size and style, there are 40 test samples, so the total size of the test set is 240. The similarity is computed with respect to images of digits with the same size and style. The mean accuracy for this method is 0.087, which makes it worse than a random classifier. The accuracy has a standard deviation of 0.021, which is relatively small, so the results are not affected much by the choice of the reference digit.

	Guided	Freehand
Large	**0.95**	**0.95**
Medium	0.875	0.65
Small	0.425	0.55

Table 2: Accuracy with dynamic time warping on trajectory method for the different combinations of sizes and writing styles.

	Guided	Freehand
Large	**0.275**	0.15
Medium	0.25	0.15
Small	0.15	0.175

Table 3: Accuracy with optical flow method for the different combinations of sizes and writing styles.

Perhaps the metric used to compute image similarity is not well adapted given that here the images are very low resolution and so some other way of computing image similarity would be more effective. Also, this method does not at all take into account the temporal dimension of the data.

4.2 Results for dynamic time warping on trajectory method

The accuracies for the different combinations of sizes and writing styles are shown in Table 2. The mean accuracy for this method is 0.733. The accuracy has a standard deviation of 0.167, so the results are also not affected much by the choice of the reference digit.

This method is much more effective than the previous one. Some additional preprocessing such as removing the beginning and start of the trajectories which are often jagged, or smoothing the trajectories could likely improve this result even more.

4.3 Results for optical flow method

The accuracies for the different combinations of sizes and writing styles are shown in Table 3. The mean accuracy for this method is 0.193, making it a bit better than the firwst method. The accuracy has a standard deviation of 0.048, so the results are also not affected much by the choice of the reference digit.

The main issue with this method is that it relies heavily on being able to match corresponding trajectories extracted from the capacitive image sequences. However, the extracted trajectories often vary a lot between samples, so it might be interesting to try computing a mean trajectory for every sample, as a way of reducing this variance.

Size	Large	Medium	Small
Mean accuracy	0.950	0.762	0.488

Table 4: Accuracy for different digit sizes.

Style	Guided	Freehand
Mean accuracy	0.750	0.717

Table 5: Accuracy for different digit writing styles.

4.4 Discussion

We have thus identified that the method using DTW on the trajectories is by far the most effective.

4.4.1 Comparison of writing parameters with DTW on trajectories method

Table 4 shows that while large and medium sized digits have decent accuracies, small digits are much more difficult to make predictions for. This is a limitation which might be tricky to overcome, given how little the palm moves as writing gets smaller and smaller.

Table 5 shows that guided digits are recognized slightly more accurately, but the difference is not too bad, which is reassuring as in practice, writing will not be guided.

4.4.2 Evolution of accuracy depending on how many digits are being predicted

To investigate how the number of classes between which we are predicting affects the accuracy of our methods, and thus how well they will scale when adding letters for example, we plot the evolution of accuracy depending on how many digits are being predicted in Fig.5.

We notice that overall, as more digits get added, the resulting decreases in accuracy gets less and less sharp. This is an indication that when adding even more characters, the method using DTW on the trajectories could still be decently effective.

4.4.3 Similarity heatmap

To further investigate how similar the center of mass trajectories are for the same digits, Fig.6 shows a heatmap of the similarity matrix for all possible pairs of trajectories. The digits are ordered, meaning that the first 30 rows/columns of the matrix correspond to the digit 0, the next 30 to the digit 1, etc. Brighter colors mean that the similarity is higher.

We notice that the highest similarity values are among the squares on the diagonal where the row and column correspond to the same digit. We can also see that digit 0 is quite similar to digit 6 and that digit 1 is a bit similar to digit 7. These observations

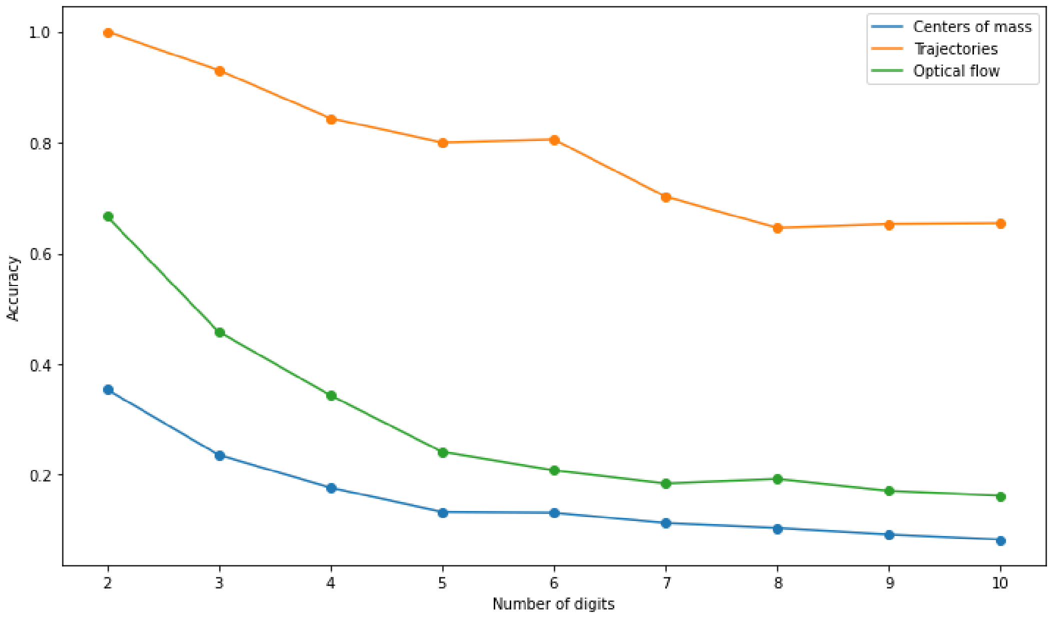

Figure 5: Evolution of accuracy depending on how many digits are being predicted.

give us more confidence that the center of mass trajectory representation is effectively regrouping sequences of capacitive images by the digit they represent.

5 Conclusion

With this project, a first step has been taken in the direction of the goal of writing on a tablet screen with a normal pen. An appropriate tablet has been found, its kernel has been modified to access its capacitive data, and an app has been created to record all the necessary data at the same time. The stage is now set for work on the handwriting recognition, with a first model which can recognize digits with an accuracy of 0.733. Insight has also been gained as to how this model is affected by the size of the handwriting, the number of different characters present, etc.

While there are still many challenges before achieving the ultimate objective of real-time freehand writing recognition, a few next steps could be useful to improve the current work.

First, the data collection is a bit tricky for now as the Leap Motion loses track of the hand quite often, meaning that the current character must be started over. This might be mitigated by adding a second Leap Motion to the setup, allowing to also track the palm from a different angle.

Furthermore, the current models could be improved by adding some more preprocessing, as detailed in [1] for example.

Finally, more advanced models such as a Convolutional Recurrent Neural Network or one using hand pose prediction as an intermediate step could be considered.

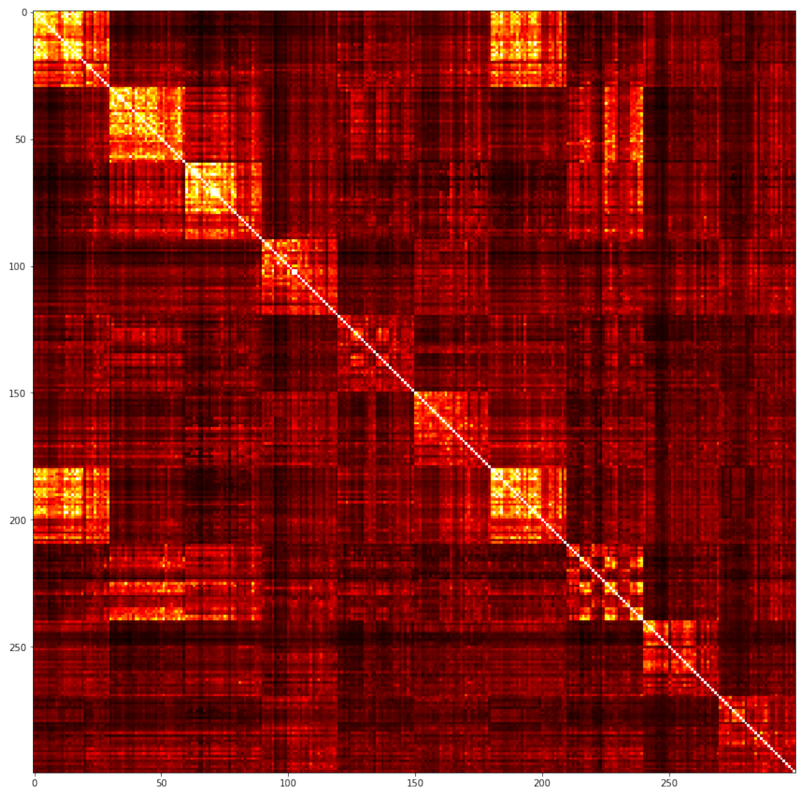

Figure 6: Heatmap of the similarity matrix for center of mass trajectories.

References

[1] C.C. Tappert, C.Y. Suen, and T. Wakahara. “The state of the art in online handwriting recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.8 (1990), pp. 787–808. DOI: `10.1109/34.57669`.

[2] Roberta H. Yuhas, Alexander F. H. Goetz, and Joseph W. Boardman. “Discrimination among semi-arid landscape endmembers using the Spectral Angle Mapper (SAM) algorithm”. In: 1992.

[3] Stan Salvador and Philip Chan. “Toward Accurate Dynamic Time Warping in Linear Time and Space”. In: vol. 11. Jan. 2004, pp. 70–80.

[4] Karan Ahuja, Paul Streli, and Christian Holz. “TouchPose: Hand Pose Prediction, Depth Estimation, and Touch Classification from Capacitive Images”. In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. UIST ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 997–1009. ISBN: 9781450386357. DOI: `10.1145/3472749.3474801`. URL: `https://doi.org/10.1145/3472749.3474801`.

[5] Frederick Choi, Sven Mayer, and Chris Harrison. “3D Hand Pose Estimation on Conventional Capacitive Touchscreens”. In: *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*. MobileHCI ’21. Toulouse amp;

Virtual, France: Association for Computing Machinery, 2021. ISBN: 9781450383288. DOI: 10.1145/3447526.3472045. URL: https://doi.org/10.1145/3447526.3472045.

[6] Thomas Berkane. *Data Analysis*. URL: https://github.com/tberkane/TouchWriteAnalysis.

[7] Thomas Berkane. *Data Collection*. URL: https://github.com/tberkane/TouchWriteDataCollec

[8] Huy Viet Le. *ACCESSING RAW CAPACITIVE IMAGES ON COMMODITY SMARTPHONES*. URL: http://huyle.de/blog/capacitive-images/.

[9] Sven Mayer. *HankIK*. URL: https://github.com/FIGLAB/3DHandPose/tree/main/device/android.

[10] *RealSense*. URL: https://www.intelrealsense.com/sdk-2/.

[11] Samsung. *S Pen Remote SDK*. URL: https://developer.samsung.com/galaxy-spen-remote/s-pen-remote-sdk.html.

[12] *UltraLeap*. URL: https://www.ultraleap.com/.

[13] LineageOS Wiki. *Info about gta4xlwifi*. URL: https://wiki.lineageos.org/devices/gta4xlwifi/.

[14] Isaac Zinda. *RainCheck*. URL: http://isaaczinda.com/raincheck/capacitance-values.html.