Recognizing Text Using Motion Data From a Smartwatch

Luca Ardüser, Pascal Bissig, Philipp Brandes, Roger Wattenhofer
ETH Zurich
Switzerland
firstname.lastname@ethz.ch

***Abstract*—In this paper, we show how motion data collected with a smartwatch can be used to infer text written on a whiteboard. All data is gathered by a commercial, off the shelf smartwatch, without any additional hardware. We obtain single letters recognition rates of 94% in the first guess. Using three guesses, we recognize 99% of the letters. This result is achieved by transforming raw sensor measurements into a coordinate system reflecting the orientation of the whiteboard. Therefore, training data recorded on one whiteboard can be used on another even if the user is writing at different heights or body postures. We show that the built in microphone picks up the sounds caused by the pen which can help to segment the input into individual letters.**

I. Introduction

Smartwatch development has been rapidly accelerating in the past few years. Nowadays, these devices are full of sensors that allow to track the movement of a user's wrist in great detail. However, most interaction methods still rely on touch or voice input. While touch input on small screens can be inefficient, voice input might be unsuitable for quiet environments, e.g., libraries or because of privacy concerns.

We present a system that can recognize letters by analyzing motion data recorded with a state of the art commercial smartwatch.[1] Generally speaking, we build a gesture recognition system and evaluate its performance on a set of 26 distinct gestures (the Roman alphabet). Letters are interesting since people do not have to artificially learn the gestures. However, also arbitrary gestures could be trained and tested. Rejecting- or accepting a phone call, muting an alarm, skipping or pausing a song are just a few of the commands that could be mapped to gestures. The similarity measure we use for this is based on Dynamic Time Warping (DTW). We focus on creating a classification method which performs well with few training samples, which is why the system was tested using 3 training samples for each of the 26 letters. The classification method does not only provide the most likely gesture but creates a ranking of gestures depending on the similarity to the input. Therefore, we could also predict the second likeliest gesture and so on. We also apply our method to written words. However, naively segmenting the data has proven infeasible. We show how the pen's sound can be used to extract isolated strokes, which greatly simplifies the segmentation problem.

[1]We used a LG Watch R for our experiments, but any modern smartwatch can easily be used.

II. Related Work

Gesture driven interaction is an extensively studied problem. The methods employed vary from pattern recognition in camera images [1], using special gloves to record the data [2], to more similar technologies to the smartwatches used in this paper, e.g., a Wiimote [3]. Consequently, smartwatches themselves have also been used for gesture recognition, e.g., in [4], [5], [6]. In contrast to these works, our system can cope with gentle gestures that only cause minor movements of the smartwatch.

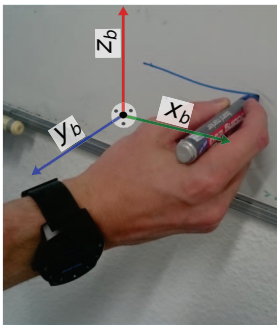
A more specialized version of gesture recognition is handwriting recognition. The most prominent solution is optical character recognition (OCR). Instead of trying to analyze the movement, the resulting image is being looked at. This process does not use any time information and is similar to how humans do it. It has a long and successful history [7]; using various techniques like hidden markov models [8] or SVMs [9].

We want to focus on approaches that use the movement of the hand. There is software to recognize written text on touchscreens [10], [11]. But since the screen is two dimensional and provides very accurate information about the x and y coordinate, techniques used in this line of work are hard to translate into our setting. Text recognition on whiteboards has been explored using infrared [12]. This approach is less easily applied than our method in everyday situations because it requires infrared transmitters and receivers attached to the whiteboard.

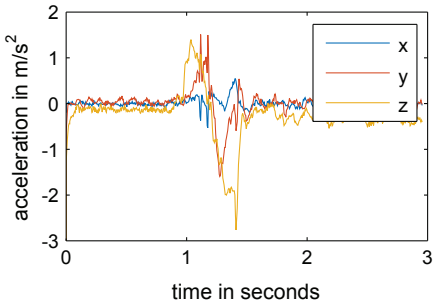
Closer to our system is research from Choi et al. [13]. They use an accelerometer directly attached to a pen to recognize written numbers. This system needs a special pen; it is evaluated using the 10 Arabic numerals, and achieves a recognition rate of more than 90%. Even though the smartwatch in our work may be seen as special hardware, it could be used for much more tasks than just recording handwriting and might become, like smartphones, an ubiquitous companion in the future.

In a recently published work Xu et. al. have also used a smartwatch-like device carried on the wrist, to realize gesture recognition [14]. They also implemented a letter recognition system which recognized up to 94.6% of the letters on average. They restricted the letters to be within a window of fixed size, while the arm has to lay on the armrest of a chair. The letters

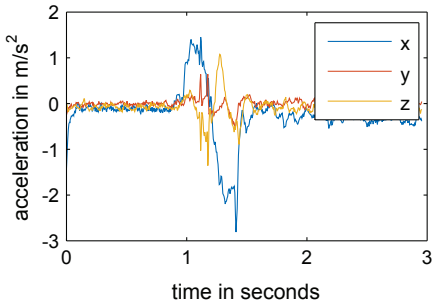
978-1-5090-1941-0/16/$31.00 ©2016 IEEE



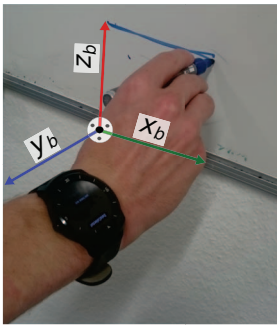
(a) clock face right



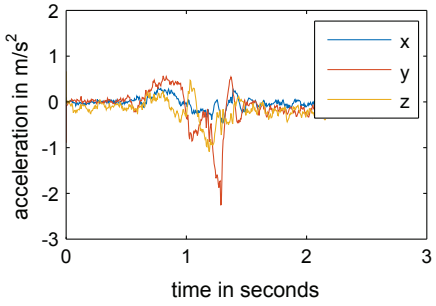
(b) Linear acceleration in device coordinates



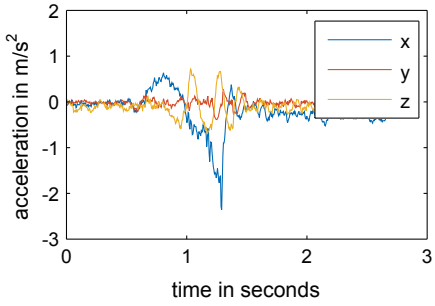
(c) Linear acceleration in whiteboard coordinates



(d) clock face up



(e) Linear acceleration in device coordinates



(f) Linear acceleration in whiteboard coordinates

Fig. 1: Drawing a straight line parallel to the x_b axis leads to different measurements depending on the watch orientation. The upper row represents the linear acceleration measurements when the clock face points to the right. The lower row corresponds to the measurements where the clock face points upwards.

were drawn using the index finger. With our system the arm position does not have to be fixed, and the letters can be written with a normal pen.

Smartwatches also have been used in an unexpected way to predict words that are typed on a keyboard [15]. The smartwatch is attached to the left wrist and depending on the movement of the wrist, a list of words that is likely to contain the written word is generated. A median of 24 words is sufficient to correctly guess the typed word. This number decreases for words that are longer than 6 letters to 10.

III. Preprocessing

The raw sensor data is influenced by changing hand postures as shown in Figure 1. Furthermore, writing at the top or bottom of the whiteboard affects these measurements. Training our system to work in all these environments is cumbersome. This would artificially increase the number of gestures we have to recognize which in general reduces the recognition performance.

To alleviate this problem, we transform raw measurements in sensor coordinates to a coordinate system relative to the whiteboard. This allows us to track user motion with respect to the whiteboard. The whiteboard coordinate system is invariant to changes in watch orientation relative to the user's wrist. Hence, we can reuse training data even if the watch is not worn exactly the same when performing the gesture. First we transform the measurements into the world coordinate system to remove the acceleration caused by the gravity of the earth. The world coordinate system consists of the z_w axis pointing to the sky, and the x_w and y_w axes lying perpendicular in the horizontal plane. No magnetometer measurements are used to relate x_w or y_w to the cardinal directions. Instead, we rely on a Kalman filter [16] to determine the z_w axis.

The z_w axis already directly relates to the whiteboard z coordinate z_b. To find x_b and y_b we assume that writing on a whiteboard mostly causes acceleration within the whiteboard plane. As a result, the x_w and y_w coordinates of the linear acceleration measurements (gravity compensated) are correlated. We use large accelerations to find the dependency caused by the whiteboard through linear regression. Figure 2 shows this correlation along with the determined whiteboard orientation. As a result we can find and track the base vectors of the whiteboard coordinate system. Since this is ambiguous, we use both possible rotations in the recognition process (the y_b axis pointing into the whiteboard and pointing out of it).

Android provides functions to transform measurements in device coordinates into world coordinates. As our results show, this transformation does not work reliably. We want to avoid to use the magnetic field sensors since errors can be introduced by magnetic fields caused by currents, magnets or magnetized material.

The recorded training gestures also contain irrelevant data before the gesture starts and after it ends. Since during training, the users are required to keep their hand still before and after

writing, we use the amplitude to segment our data. An example of this can be seen in Figure 3.

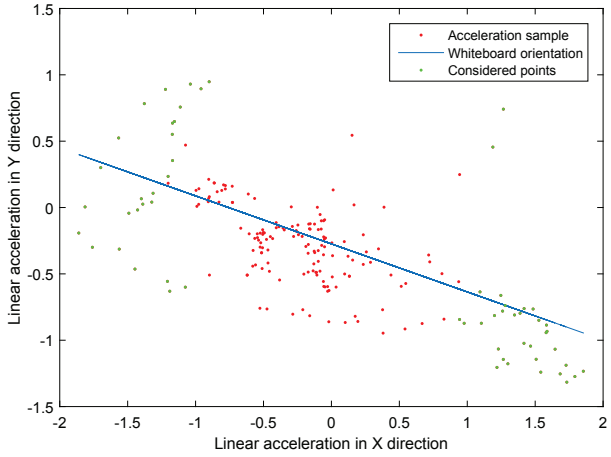


Fig. 2: The variance of the linear acceleration is large with respect to both axes. The red line shows the X-axis. The blue line represents the orientation of the whiteboard. Only the green points where considered to fit the blue line

The data is recorded during a short time interval. The time from the beginning of this window up to the moment when the user starts to write is called *left-margin*. Similarly, the time interval between the point where the user finishes writing and the end of the recording period is called *right-margin* (see Figure 3). These margins should contain no movement of the pen. The measurements which correspond to the gesture we are interested in, lay between these two margins. To eliminate these margins and extract only the relevant part of the gesture, a segmentation algorithm is used. The algorithm relies on changes in amplitude of the acceleration signal to find the start and the end point of the gesture. The amplitude corresponds to the euclidean norm of each sample of the sequence. If the amplitude is larger than a given threshold, we assume that there was pen movement. The first time the amplitude is higher than the threshold will be interpreted as the starting point p_s. If there is a longer period of time, after detecting such a p_s, in which the amplitude is continuously lower than the threshold, this will be interpreted as the end of the gesture and an endpoint p_e will be set. Extracting the sequence of measurements between p_s and p_e eliminates the margins. In addition, we want to eliminate noise, which is induced by the user before writing a letter, e.g., if he moved his hand before the application told him to do so. To achieve this, the algorithm does not stop after finding the first p_s, p_e pair, but it also considers the rest of the sequence. If there is a new value larger than the threshold, the algorithm rejects the already collected start-/endpoints and chooses the new interval. This removes noise in the beginning, but may still produce wrong segmentation values if there is noise in the end. Note that the collected data seems to indicate that noise appears mainly in the beginning.

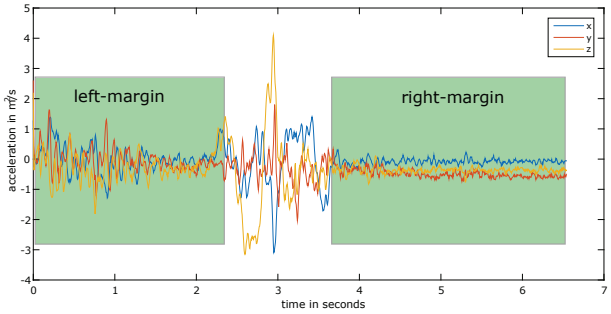


Fig. 3: Linear acceleration before the segmentation. The margins of the unsegmented data are highlighted in green. There is also a lot of noise in the left margin which is not considered by the segmentation algorithm.

A. Classification

The classification algorithm for letters consists of three stages. The first stage extracts features from the collected data. We evaluated various features of which the following performed best:

- **WhiteboardCoords** Gyro & linear acceleration data transformed into the whiteboard system as described above.
- **SensorCoords** Linear acceleration & gyro data in sensor coordinates.
- **SensorCoordsAndroid** Linear acceleration & gyro data provided by Android in sensor coordinates.

In the second stage, the sequences of feature vectors are compared using the DTW algorithm [17]. To compare two feature vectors, we use the euclidean distance.

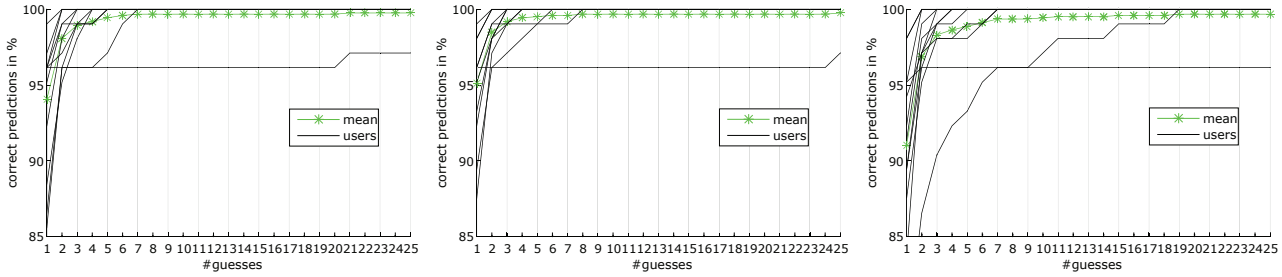
In the third stage we create a ranking out of the similarity scores between test- and training-letters. This ranking captures the similarity between the input and all the training samples of all letters.

IV. Evaluation

Recall that the recorded data depends on the orientation of the watch because the device may change its orientation depending on the height the user writes on the board. Therefore, we implemented the transformation into the whiteboard coordinate system as described earlier in order to obtain orientation independent acceleration measurements.

Figure 5 shows the difference between non-transformed and transformed data. In this case the whole alphabet was written by one of our users above the lower edge of the whiteboard and at the upper edge twice. The gestures written at the top are compared to the gestures written at the bottom and vice versa. We used the features WhiteboardCoords, SensorCoords, and SensorCoordsAndroid. We test two gestures of each letter against two training gestures for each letter. As we can see, we get the lowest performance using the linear acceleration provided by Android. In the remainder, both the test and the training set were written on approximately the same height.

Twelve people were asked to write each letter of the alphabet four times. The test subjects were asked to write at



(a) WhiteboardCoords (b) SensorCoords (c) SensorCoordsAndroid

Fig. 4: The recognition performance for each feature. The plots show how many letters were correctly recognized depending on the number of guesses available. After 3 guesses WhiteboardCoords correctly recognizes 99% of the letters. Even though this is slightly worse than SensorCoords, this feature is superior since it can also handle letters written on a different height (see Figure 5).

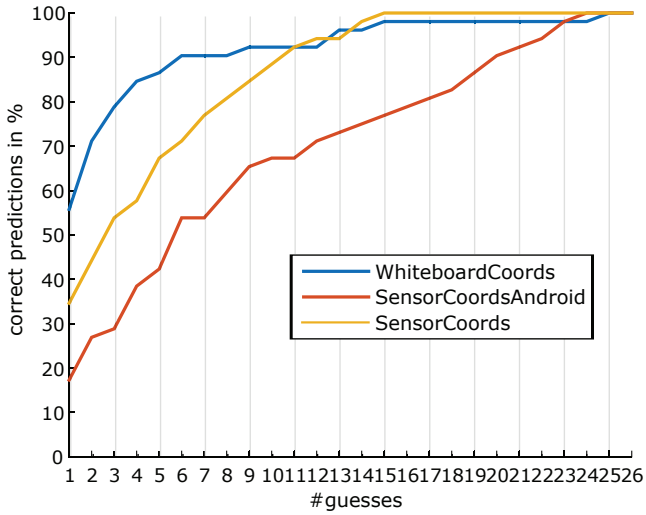


Fig. 5: The effect of transforming the data into whiteboard coordinates. The test gestures were written at the upper edge while the train gestures were written at the lower edge of the board. The transformation improves the result.

a comfortable height without further restrictions. We test the algorithm offline by using three samples of each gesture as training set and one for testing with cross validation.

A. Letter Recognition

In Figure 4 the performance for all users is shown. Using our rotation into whiteboard coordinates slightly reduces the performance to about 94%, but it allows us to use the training data on different whiteboards. Within three guesses, we can predict the correct letter in 99% of the cases. The percentage of letters that was predicted correctly after a specific number of guesses is shown for each user. This shows that the rotation into whiteboard coordinates clearly outperforms the linear acceleration provided by Android.

B. Letter Confusion

Figure 6 shows the average similarity rank for each test letter (Y-Axis). The diagonal values have the best (lowest) entries. This means that our system predicts low ranks for the correct letters on average. The entries for the four letter pairs (“I”,“J”), (“M”,“N”), (“E”,“F”) and (“U”,“V”) shows that these pairs are easily confused. This makes sense, since these letters look somewhat similar. Generally, the letter “E” can be distinguished well. This can be explained by the relatively long time and the number of strokes it takes to write an “E”. Note that this matrix does not need to be symmetric. We take the column of the letter “E” as an example. “E” appears late in the ranking for many other letters but this does not mean that all letters appear late in the ranking of an “E”. Many of these letters will produce high costs comparing them to an “E”.

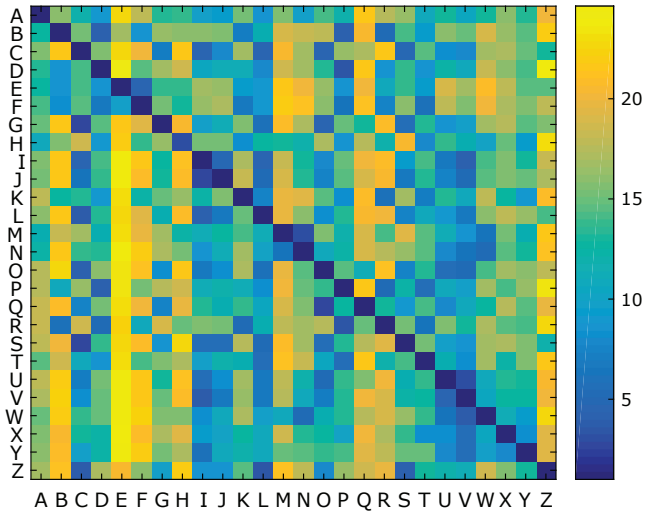
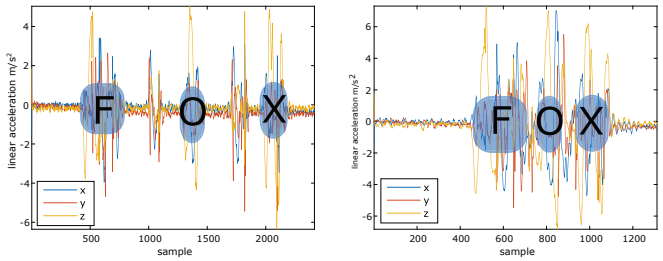


Fig. 6: This matrix represents the average ranking for each letter and all users if we use the WhiteboardCoords feature. The test letters are shown on the Y-Axis, the values on the X-Axis represent the ranking.

C. From Letters to Words

In order to extract letters from unsegmented sensor recordings, we need to find starting points of gestures. This problem

is illustrated in Figure 7. If the user writes quickly, then segmenting the individual letters is non-trivial.



(a) The word "FOX" written slowly (b) The word "FOX" written quickly

Fig. 7: Finding the start and the endpoint of a letter is difficult if the word is written quickly.

First, we naively assume that each sample in the recordings is a starting point and generate a ranking of letters for it. However, finding the starting points of gestures from this sequence of rankings has proven to be difficult. Figure 8 shows that the DTW distances do not reliably drop significantly when a gesture starts. Hence, this method does not work directly. Furthermore, for most (random) time segments, there is a trained letter that matches the sensor measurements in said segment. This means that in order to find as many correct gestures as possible, we have to accept a massive number of false positives caused by the difficulty to segment the input correctly.

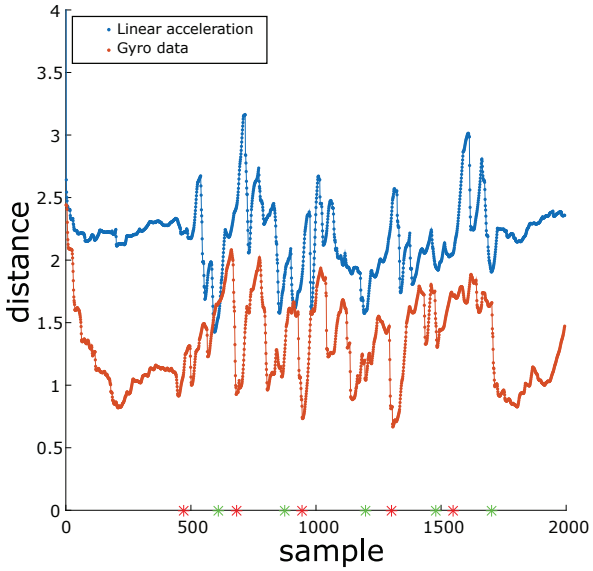


Fig. 8: Distance for each feature of the first guess of every evaluated point in time when using a naive approach on the word "JUMPS". The red and green marks show the start and end points of the letter according to the ground truth.

To reduce the number of false positives, we segment the sensor measurements using audio recorded with the smartwatch while writing. When a pen is dragged on the whiteboard surface, it emits a distinct sound that can be used to track isolated strokes on the whiteboard.

Figure 9 shows the spectrogram of an audio track recorded when writing "OVER". The strokes of each letter are easily visible. Our audio assisted recognition algorithm uses this information and only applies our DTW distance metric whenever a stroke starts. We call the starting points of strokes *Points of Interest* abbreviated as POI.

The detection algorithm sums up the absolute amplitude value of all frequencies between 5 kHz to 10 kHz for each point in time. This gives us a vector $\mathbf{p}$ where $\mathbf{p}_i$ contains the sum for the i^{th} point in time. We then search for values in $\mathbf{p}$ which are greater than a threshold value τ. Multiple successive values greater than τ will be combined.

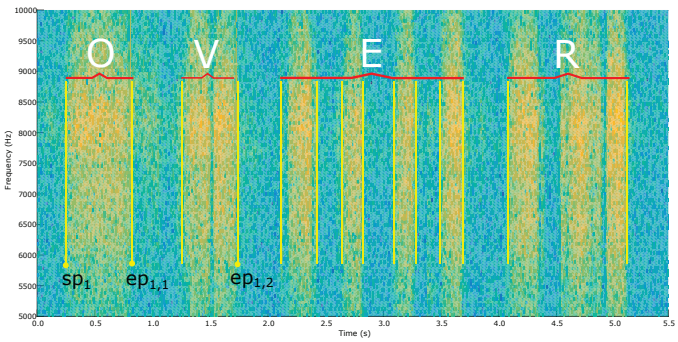


Fig. 9: Spectrogram of the word "OVER" with highlighted stroke start- and endpoints.

Note that the recording of the audio and the recording of the acceleration do not start exactly at the same time. Because of that, we align the two signals using the cross correlation between $\mathbf{p}$ and the evolution of the magnitude of the linear acceleration. An example is shown in Figure 10. Now, we are able to extract the starting and ending of a stroke with respect to the linear acceleration signal.

We run our DTW distance metric for every POI. Because letters may consist of multiple strokes, we cannot use the next stroke endpoint to extract the test sequence. Instead, we use a time window of length 2.5 seconds to select possible endpoints. As a result, every starting point leads to multiple test sequences, which are individually passed to the recognition stage.

In total, eight users wrote 428 words. Half of these words were written with a small break between the letters; the other half without any restrictions. The ground truth was manually annotated. We evaluated the performance using the WhiteboardCoords feature, which performed very well in the letter recognition task and also works for arbitrary writing heights.

The naive approach, considering each sensor sample as a possible starting point for a letter, correctly recognized 64.5% of the letters within 3 guesses. However, the method produces nearly 250000 erroneous predictions. Most of them in places no letter started.

Using the audio signal to reduce the number of possible starting points of gestures increased the recognition rate to 71.2% within three guesses. It also allowed us to reduce

the number of erroneous predictions by nearly two orders of magnitude.

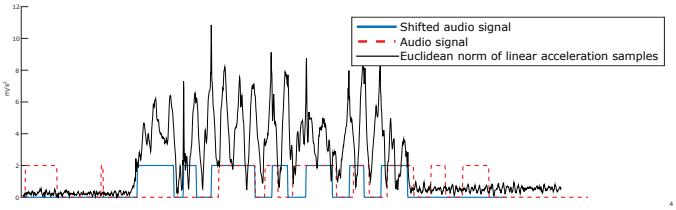


Fig. 10: Cross correlation between an audio signal and the corresponding acceleration measurements.

V. Discussion

Our experiments show that recognizing single letter works reliably. This holds even if the training set is very small as it is in our case. Hence, our system could be used for simple commands, e.g., cycle through slides while writing on a whiteboard. Since a real world system could continue to learn by adding correctly classified gestures to the training set, its performance would be even better. One can also imagine a user writing words if she takes a small pause between every letter; even though we admit that using OCR software will most likely yield better results at the moment.

Recognizing letters within words is possible to some extent if we know when the letters appear, i.e., the start and end time. However, without solving the segmentation problem, our approaches lead to prohibitively large numbers of erroneous recognitions. Using the audio data helps reducing the number of false positives dramatically and is a first stepping stone. Furthermore, since we know when a gesture ends, it even increases the recognition rate. Hence, using audio is superior to the naive approach. We imagine that with significantly more training data, error rates should be lower. Once the false positive rates go down far enough, one can apply a dictionary matching algorithm. Similar to software keyboards or T9, such an algorithm can estimate the most likely word even based on ambiguous letter input.

VI. Future Work

There are still open problems and performance issues that could be addressed in the future. Having significantly more training and test data available, might allow to fine tune parameters used for the DTW metric.

Using the audio data reduced the amount of false positives dramatically while simultaneously increasing the number of true positives but our method is only reliable in a silent environment. Perhaps one could enhance this method by making it more robust against noisy environments.

If there is a way to choose the most probable letter, other possible predictions may drop out which would reduce the amount of false positives. This may happen because if we decide a letter ℓ has been written at a point in time, then another letter could not be written within the duration of the letter ℓ. Therefore, all predictions within this time interval may be ignored. Another open problem is how to detect spaces between words or how to segment different words from each other. One could introduce a special gesture which stands for a space. There may be an extension of this work solely focusing letters written on different heights. In addition to fine tuning our approach, we also envision a system that uses the barometer and the sensors used in this paper, to estimate the relative height in which words were written.

References

[1] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International workshop on automatic face and gesture recognition*, vol. 12, 1995, pp. 296–301.

[2] R.-H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE, 1998, pp. 558–567.

[3] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*. ACM, 2008, pp. 11–14.

[4] G. Raffa, J. Lee, L. Nachman, and J. Song, "Don't slow me down: Bringing energy efficiency to continuous gesture recognition," in *Wearable Computers (ISWC), 2010 International Symposium on*. IEEE, 2010, pp. 1–8.

[5] G. Bieber, T. Kirste, and B. Urban, "Ambient interaction by smart watches," in *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2012, p. 39.

[6] L. Porzi, S. Messelodi, C. M. Modena, and E. Ricci, "A smart watch-based gesture recognition system for assisting people with visual impairments," in *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices*. ACM, 2013, pp. 19–24.

[7] R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 63–84, Jan 2000.

[8] H. Bunke, M. Roth, and E. G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden markov models," *Pattern recognition*, vol. 28, no. 9, pp. 1399–1413, 1995.

[9] C. Bahlmann, B. Haasdonk, and H. Burkhardt, "Online handwriting recognition with support vector machines-a kernel approach," in *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*. IEEE, 2002, pp. 49–54.

[10] D. Dutta, A. R. Chowdhury, U. Bhattacharya, and S. Parui, "Lightweight user-adaptive handwriting recognizer for resource constrained handheld devices," in *Proceeding of the workshop on Document Analysis and Recognition*. ACM, 2012, pp. 114–119.

[11] W. Kienzle and K. Hinckley, "Writing handwritten messages on a small touchscreen," in *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*. ACM, 2013, pp. 179–182.

[12] M. Liwicki and H. Bunke, "Handwriting recognition of whiteboard notes," in *Proc. 12th Conf. of the Int. Graphonomics Society*, 2005, pp. 118–122.

[13] S.-D. Choi, A. S. Lee, and S.-Y. Lee, "On-line handwritten character recognition with 3d accelerometer," in *Information Acquisition, 2006 IEEE International Conference on*. IEEE, 2006, pp. 845–850.

[14] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 9–14.

[15] H. Wang, T. Tsung-Te Lai, and R. R. Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 155–166.

[16] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[17] "Dynamic time warping," in *Information Retrieval for Music and Motion*. Springer Berlin Heidelberg, 2007, pp. 69–84.