SmartWatch for Wall Writing: Real-time Transcription of Wall Writing from Inertial Sensing

Snigdha Das, Satyam Awasthi, Abdul Shamnar P, Pradipta De, Sandip Chakraborty, Bivas Mitra

***Abstract*—This paper presents a smartwatch-based application, called *WatchScribe*, to convert any wall writing to virtual boardworks. Users can scribe anything over this virtual whiteboard with zero-bootstrapping by mimicking the writing on the wall with a pen. Transcribing wall-writing with a smartwatch is challenging because of two reasons. (a) As the smartwatch's orientation changes continuously due to wrist movements, the pen's locus is different from that of the smartwatch. (b) There are events when the user lifts the pen from the whiteboard and starts at a different position to draw the next stroke. *WatchScribe* leverages locomotive data from a smartwatch to estimate the pen's locus using an unsupervised method. Further, it uses a light-weight approach to extract the writing micro-gestures from the estimated pen's locus and reproduces a real-time transcription of the wall-writing. We apply qualitative human-driven and novel quantitative metric-based evaluation techniques for checking the correctness of *WatchScribe* over the data collected from 10 participants in a very diverse setup. Compared to the closest baseline that uses a smartphone's corner as a pen, we achieve an average 9% reduction in disparity score and 27% improvement in average accuracy in terms of decipherability of the reproduced transcripts.**

***Index Terms*—smartwatch; virtual whiteboard, free-form writing, wall-writing**

I. Introduction

During the COVID-19 pandemic situation, the teaching mode has been majorly shifted from a physical classroom to a virtual one through various interactive meeting platforms. For running an online classroom where the students and teachers are not co-located, the first requirement is to set up an online teaching infrastructure that can replace a physical classroom environment. Such infrastructure includes devices like pen-tablet or touchscreen-based computers that can replace a physical whiteboard. Apart from the classrooms, whiteboards are also indispensable and are widely used in various business and corporate meetings. However, a touchscreen-based system or a pen-tablet might not be readily available to every individual associated with such a profession. A policy report by United Nations concerning the education during COVID-19[1] have indicated that the pre-primary and primary teachers typically lack the necessary digital skills and even need technician's support to set up a pen-tablet. For example, the report indicates that only 64 percent of primary and 50 percent of secondary teachers in sub-Saharan Africa have received a minimum training, which often does not include necessary digital skills.

Considering the rapid penetration of smartwatches across every corner of the globe [1], in this paper, we explore whether we can convert any room-wall to a virtual whiteboard with the help of a smartwatch having simple inertial sensors. Regular smartwatches are typically cheaper but have wider usage compared to specialized systems such as pen-tablets. Therefore, a virtual whiteboard application using a smartwatch provides significant value addition to the device. The core idea of developing such a system would be to track the arm movement using the inertial sensors embedded with the smartwatch for estimating the locus of the pen used for writing virtually on the wall. The locus of the pen can then be used for real-time continuous reproduction of the boardwalk. The pen's locus can thus directly be projected on the wall for having a realistic feeling of writing over a virtual whiteboard (as shown in Fig. 1), or can be streamed directly to other audiences. The primary advantage of using *WatchScribe* is that it does not require particular digital skills for using a specialized system like a pen-tablet. The teachers or the presenters can directly write on a vertical surface that they are already comfortable with. However, the major challenge in this task is to devise a complete unsupervised zero-bootstrapping approach for estimating the pen's locus from the arm movement, as practically anything in any language can be scribbled on a whiteboard. Furthermore, it can be noted that the pen's locus is different from the arm or the wrist movements, whereas the inertial sensors attached with a smartwatch can only capture the arm or the wrist movements.

Several recent research works have focused on tracking the arm movement from inertial sensing data [2]–[6]. Many of these works have also used the smartwatch data for handwritten text identification [2], [7]–[13] or sign-language interpretation [14], [15]. However, capturing unconstrained scribbles during a boardwork requires an altogether different approach from the existing ones. Except [2] and [13], the proposed methods primarily work on a fixed set of characters for a fixed language; therefore, they use pre-trained supervised models to recognize the characters from the raw sensor data. However, a user can scribe anything on a whiteboard, either texts or symbols or figures, or any combination of them. Therefore, a supervised model fails to recognize characters in those scenarios correctly. Although the approach proposed by Yin *et. al.* [13], called AirContour, has an unsupervised component for wrist-trajectory generation from the smartwatch inertial sensor data, the representation of the generated trajectories in the form of a meaningful character still uses a supervised method. Further, both AirContour and ArmTrack [2] were developed for on-air writing using a smartwatch. Therefore, these approaches can compute continuous trajectories only, which are

[1]https://www.un.org/development/desa/dspd/wp-content/uploads/sites/22/2020/08/sg_policy_brief_covid-19_and_education_august_2020.pdf (Access: October 1, 2021)

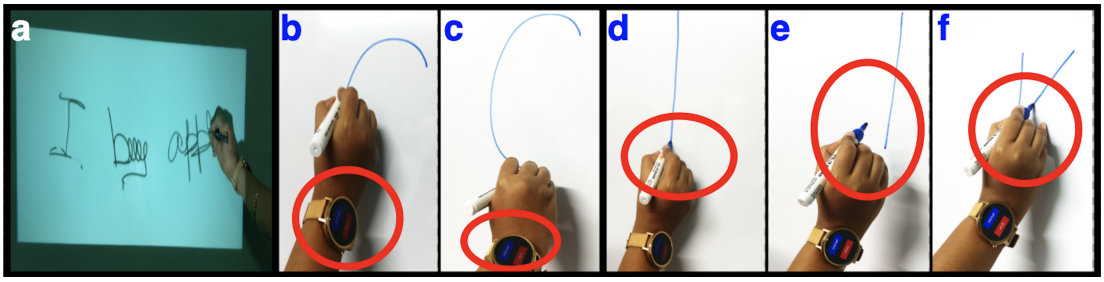


Fig. 1: (a) *WatchScribe* in action; (b) – (f) Challenges in physical whiteboard capture using a smartwatch: (b), (c) indicate that the locus of the pen is different from the locus of the smartwatch; also the orientation of the smartwatch changes during the course of writing, (d) shows the first stroke of a multi-gesture character 'K', (e) shows the pen-up operation during the writing, (f) indicates the moment when the pen touches to the board again to continue the next stroke of 'K'

not suitable for unconstrained scribble representations, for the reasons explained below.

As we mentioned earlier, one must track the pen's locus during the boardwork, based on the inertial sensing data from the smartwatch. However, smartwatch sensors attached to the wrist are limited in their precision in generating such a locus. Fig. 1 illustrates the challenges in designing such a system. First, the locus of the pen and the locus of the wrist-worn smartwatch are not aligned. Therefore, the gesture-trajectories computed based on the smartwatch's inertial sensors differ from the pen's locus. Besides, the smartwatch orientation keeps on changing during the boardwork, depending on the factors like the curvature of the text, height of the user versus height of the text, size of the text, and so on. Hence, a simple trajectory generation from the smartwatch's inertial sensing data does not work in this case. Second, as shown in Fig. 1(d)–(f), any boardwork scribble is a combination of multiple small free-form strokes of basic shapes (like lines, curves, etc.), which we call *micro-gestures*. A script can have a single micro-gesture (for instance, the English character *'C'*) or multiple micro-gestures (say, English character *'K'*, composed of three micro-gestures). Multiple micro-gestures come as a sequence of first drawing the line, then a quick act of lifting the pen and moving to the next starting point to continue, designated hereafter as the *pen-up/pen-down (PUPD) micro-gesture*. Approaches like AirContour [13] and ArmTrack [2] fail to capture such PUPD micro-gestures. To correctly capture the boardwork scribbles, the micro-gestures must be faithfully identified based on the pen's estimated locus.

Owing to these challenges and the limitations of the existing works, we develop *WatchScribe*, which aims to capture any physical boardwork in real-time by accurately identifying various micro-gestures using smartwatch data (Section II) and thus can project them on a virtual whiteboard in a real-time. The system does not require any bootstrapping since we use a complete unsupervised approach without any pretraining. Further, it is a usable one-shot system that does not require any prior writing practice using the device. We use locomotive data from the smartwatch to calibrate the equipment towards a fixed orientation and then estimate the locus of the pen by computing the locus of the smartwatch and projecting it over the 2D plane of the wall that acts as the virtual whiteboard (Section III). To eliminate the PUPD micro-gestures, we develop an intelligent approach based on clustering the sensor data and then identify the micro-gestures corresponding to the boardworks (Section IV).

WatchScribe has been implemented as an Android application over a Moto-360 smartwatch. We have carried out thorough experiments with 10 participants including both right and left-handed with boardwork scribbles using the alphabets, digits, and mathematical symbols from two different languages – English and Bengali (total 133 characters), and 36 carefully chosen words from each of the two languages (for English, the first character in both uppercase and lowercase, so a total of 72 words). We develop a novel quantitative metric-based technique followed by a qualitative human-centric method for evaluating the decipherability of the transcripts generated by *WatchScribe* (Section V). Relative to the closest baseline [16], we achieve an average 9% reduction in the disparity score from the metric-based quantitative evaluation and a 27% improvement in average decipherability accuracy from the human-centric evaluation (Section VI). *WatchScribe* takes overall 0.52 seconds median running time with a standard deviation of 0.29 seconds, which demonstrates the suitability of *WatchScribe* to generate real-time transcripts (Section VI).

II. System Overview

Fig. 2 shows the overall architecture of *WatchScribe*, which is primarily composed of two modules: (a) *Pen Locus Estimation*, and (b) *Transcription Generation*. *WatchScribe* gets the locomotive (accelerometer & gyroscope) data from a smartwatch, processes and generates a real-time transcription of the boardwork. The continuously generated transcript can be projected on the wall to provide a virtual whiteboard environment to the teacher or presenter and can also be streamed simultaneously to the devices (laptop, desktop, or mobiles) of other participants or observers.

(a) Pen Locus Estimation: This module estimates the locus of the pen from the locomotive (accelerometer and gyroscope) data captured from the inertial sensors embedded over the smartwatch. First, the sensor stream is pre-processed to eliminate any noise. Next, we apply an Euler angle-based

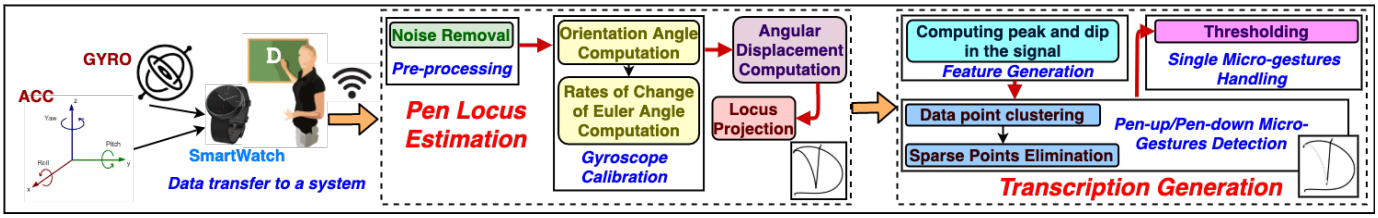


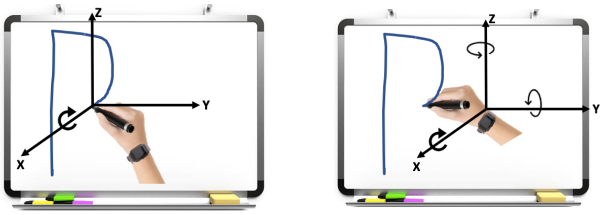
Fig. 2: *WatchScribe* System Architecture showing the on-device pen locus estimation & transcription generation from the smartwatch inertial sensing.

method [17] to correct the frame of reference (FoR) of the smartwatch to calibrate the gyroscope readings. Finally, we estimate the pen's locus from the calibrated gyroscope readings. Notably, the estimated pen's locus comprises of (i) single or multiple micro-gestures, most of which correspond to the written scripts, as well as (ii) the pen-up and the pen-down (PUPD) events (Details in **Section III**).

(b) Transcription Generation: This module generates the final transcriptions by extracting the relevant micro-gestures from the pen's locus estimations. Boardwork transcription requires putting together multiple micro-gestures after eliminating any PUPD micro-gesture. We identify the PUPD activities leveraging the sensor signatures, which can mark those events. Since the locus between the pen-up and pen-down does not lead to any content, this module discards these micro-gestures to generate the final transcription (Details in **Section IV**).

III. Estimating Pen Locus

As shown in Fig. 3a, any boardwork can be transcribed to its digital form by continuously estimating the locus of the pen. Although the locus can be estimated by continuously computing the relative displacement of the pen-tip from its acceleration, this process is challenging when the locomotive data is captured from the wrist-worn smartwatch. The reasons follow. (1) Accelerometer gives the proper acceleration with respect to an instantaneous rest frame, which is different from the coordinate acceleration. Incidentally, we need to compute the displacement of the pen-tip with respect to the virtual whiteboard, and thus need the coordinate acceleration, which is the acceleration with respect to a fixed coordinate system (here the coordinates of the virtual whiteboard). (2) The existing approaches for estimating the coordinate acceleration from the proper acceleration are known to be error-prone [18], and thus can not be applied to compute displacements minutely on a fine scale, which is required in our case for properly decoding individual characters of the written text. An alternate approach to solve this issue is to consider the fact that the translation displacement on a 2D plane is equivalent to the angular displacement across the axis perpendicular to that plane [13]. For example, in Fig. 3a, we observe that the pen's locus over the y-z plane (the vertical plane of the board) can be calculated from the angular displacement along the x-axis (we call this axis as the primary axis for the boardwork). Notably, this method can only be applied when the sensor is mounted at the pen-tip, such that the direction of the angular



(a) Sensor on the pen-tip (b) Sensor on the smartwatch

Fig. 3: Computing the locus of the pen: (a) the angular drift along x-axis gives the locus of the pen-tip. (b) As the smartwatch orientation changes due to the wrist movement, all the axes undergoes rotation.

displacement fits along the x-axis. However, Fig. 3b indicates that the smartwatch orientation gets changed continuously during boardworks, due to the flexion and extension of the wrist [7]. Consequently, the gyroscope on the smartwatch experiences angular displacements across all three axes. This makes the locus estimation of the pen on a fixed 2D coordinate system challenging when the data is captured over a wrist-worn smartwatch.

To solve this problem, intuitively, we have to calibrate the gyroscope readings towards a fixed coordinate system to eliminate the unintended rotational drifts across all the axes except the primary axis for the boardwork. For this purpose, we first pre-process the raw locomotive data for external noise filtering and then apply a rotational kinematics-driven approach for gyroscope calibration. Let $\overrightarrow{\mathbf{ra}} = [ra_x, ra_y, ra_z]$ and $\overrightarrow{r\boldsymbol{\omega}} = [r\omega_x, r\omega_y, r\omega_z]$ be the raw accelerometer and gyroscope readings from the smartwatch, where ra_x, ra_y, ra_z denote the accelerometer readings, and $r\omega_x$, $r\omega_y$, $r\omega_z$ denote gyroscope readings, at the x, y and z axes, respectively. The details follow.

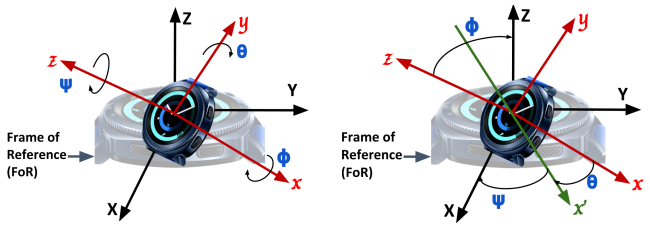
A. Data Pre-processing

We first apply a noise removal step to ensure that only the sensor data corresponding to the relevant boardwork is extracted. We capture the raw sensor data at the maximum sampling rate supported by the smartwatch for higher fidelity. However, this makes the data noisy, as even a minor arm movement can lead to a variation in the sensor readings. Since the major signal components corresponding to the boardwork events are in low-frequency domain, we apply a low pass filter with a predefined passband frequency $\mathcal{F}$ (empirically fix

$\mathcal{F} = 2Hz$) along all axes for removing the noise and obtain filtered streams $\overrightarrow{\mathbf{a}}$ and $\overrightarrow{\boldsymbol{\omega}}$, from the raw sensor data stream $\overrightarrow{\mathbf{ra}}$ & $\overrightarrow{r\boldsymbol{\omega}}$, respectively.

B. Gyroscope Calibration

In order to calibrate the gyroscope readings with respect to a fixed coordinate system, we consider the Earth's reference frame as the frame of reference (FoR) for all the computations. To eliminate the additional angular drifts that arise due to the flexion and extension of the wrist, *WatchScribe* continuously computes the change in the orientation angle with respect to the FoR, and then projects the gyroscope readings on the FoR. The detail follows.



(a) Rotation across each axes (b) Euler Angle Computation

Fig. 4: Adjusting smartwatch orientation towards the Earth's reference frame, 'XYZ' represents the axes of the Earth's reference frame and 'xyz' represents the current axes of the smartwatch.

1) Orientation Angle Computation: For estimating the smartwatch orientation concerning the FoR, we first compute the Euler angles [17], specifically the Roll (ϕ), Pitch (θ) and Yaw (ψ) angles. Fig. 4a shows these angles, where the horizontal position of the smartwatch (XYZ space) is considered the FoR for the sensors. To compute the Euler angles, we follow the 'xyz' rotation sequence, as shown in Fig. 4b. Here we first give a rotation with angle ϕ about the x-axis (which reorients the y-z plane), followed by the rotation with angle θ about the y-axis (which reorients the x-z plane, axis x changes to x'), and finally the rotation with angle ψ about the z-axis (which reorients the x-y plane, projects axis x' over X). Now, to compute these angles, we take the help of the filtered accelerometer readings across the three axes ($\overrightarrow{\mathbf{a}} = [a_x, a_y, a_z]$), as the translational motion at one plane (say, yz) can be used to compute the rotational drift across the primary axis (say, x) of the boardwork and vice versa. However, using the proper acceleration readings across the three axes of the accelerometer, we can compute only any two of these three angles by fixing the gravitational acceleration towards the Z-axis of the FoR. We choose to calculate the values of ϕ and θ, as these two angles get the maximum impact due to the gravitational acceleration, which shows a prominent signature in the accelerometer readings. Following Fig. 4b and the standard rotation matrices for Euler angle computation [17], we compute ϕ and θ from $\overrightarrow{\mathbf{a}}$ as follows[2],

$$\phi = atan2(a_y/a_z), \quad \theta = atan2(-a_x/\sqrt{a_y^2 + a_z^2})$$

2) Computing Rates of Change of the Euler Angles: Next, we compute the third angle ψ from the estimated ϕ & θ and the gyroscope readings, by applying the kinematic relation used in [19]. Essentially kinematic relation provides the *rates of change of Euler angles* $\acute{\mathbf{\Omega}} = [\acute{\phi}, \acute{\theta}, \acute{\psi}]$, which indicate the rates of change in the angular displacements with respect to the Earth's reference frame. Following [19], we compute $\acute{\mathbf{\Omega}}$ from the previously computed ϕ and θ, along with the filtered gyroscope readings $\overrightarrow{\boldsymbol{\omega}} = [\omega_x, \omega_y, \omega_z]$ as below.

$$\acute{\phi} = \omega_x + (\omega_y sin\phi + \omega_z cos\phi)tan\theta$$
$$\acute{\theta} = \omega_y cos\phi - \omega_z sin\phi$$
$$\acute{\psi} = (\omega_y sin\phi + \omega_z cos\phi)/cos\theta$$

C. Computing Angular Displacement to Estimate the Pen Locus

Finally, we estimate the locus of the pen by calculating the angular displacement from the rates of change of Euler angles $\overrightarrow{\mathbf{\Omega}}$. Precisely, we compute the angular displacement vector $\mathbf{G}_t$ by integrating the rates of change of the Euler angles at any time instance $t \in (0, T)$. For numerical integration, we apply the Trapezoidal rule as $\mathbf{G}_t = \mathbf{G}_{t-1} + \acute{\mathbf{\Omega}}_t * \delta t$, where, $\mathbf{G}_t$, $\acute{\mathbf{\Omega}}_t$, and δt represent the angular displacement vector at instance t, rates of change of the Euler angle vector at instance t, and the interval between two subsequent time instances, respectively. Since the start of the writing on the board denotes no orientation change (as we are considering the Earth's reference frame as the FoR), we initialize the starting point $\mathbf{G}_t$ at $t = 0$ as $[0, 0, 0]$.

D. Projecting the Locus on the Vertical Plane

The final task is to project the computed locus on a vertical plane, to mimic the boardwork in a digital form. Ideally, as shown in Fig. 4a, we should align the z-axis of the smartwatch with the Z-axis of the Earth's reference frame (FoR), while computing the orientations. However, during a boardwork, the smartwatch typically remains in a vertical position where, in general, the x-axis of the smartwatch gets aligned with the Z-axis of the Earth's reference frame (FoR). To fix this alignment at runtime, we identify the axis of the smartwatch (say, x-axis), which experiences the gravitational acceleration component and aligns that axis towards the Z-axis of the FoR. This allows us to project the locus of the smartwatch on the Y-Z plane of the FoR.

IV. Transcription Generation

Once we trace the locus of the pen, our next task is to discriminate the micro-gestures reflecting the actual writing vis-a-vis the PUPD events. We first extract the required features to characterizes PUPD events and then isolate PUPD

[2]The detailed calculations can be done following the rotation angle computations as discussed in [17].

micro-gestures from the continuous micro-gestures to generate the final transcription.

A. Features for PUPD Micro-gestures

While writing a character having multiple micro-gestures, a user typically performs the following actions in sequence – (1) draw a stroke, (2) lift the pen up from the board and put it at the starting position of the next stroke, (3) draw the next stroke. As lifting of the pen and moving it to a new starting position are performed in a coordinate plane, which is different from the writing plane (the plane of the virtual board), we typically observe a sharp deviation in the sensor data in the axes other than the primary axis for the boardwork. However, we cannot use a simple threshold-based approach to compute such deviations, as the absolute value of the acceleration readings depends on multiple factors like the arm posture, the pressure applied on the wall while writing, cursiveness of the text, etc. Therefore we use an intelligent and novel approach by applying a hierarchical clustering-based approach over the gyroscope data, as follows.

Let P_k^i and D_k^j denote a peak (a local maximum) at time instance i, and the immediately next dip (a local minimum) at time instance j, respectively, on the gyroscope readings across the axis k where $k \in \{x, y, z\}$. Let $d_k^{(i,j)} = (P_k^i - D_k^j)$ be the local deviation from the gyroscope readings for the axis k. Interestingly, the user might rotate his arm while performing the PUPD actions; therefore, the dip corresponding to a peak can also be observed on two different axes. Therefore, we also compute the inter-axes deviation as $\hat{d_k}^{(i,j)} = (P_k^i - D_{\bar{k}}^j)$, where k and $\bar{k}$ are two different axes. Let $\mathbb{D}$ be the set of all these deviation values, both intra-axis deviations and inter-axes deviations. Next, we need to find out which of these deviation values correspond to the micro-gestures correspond to the actual writing and the PUPD micro-gestures.

B. Detecting PUPD Micro-gestures

To detect PUPD micro-gestures, we implement k-means clustering on $\mathbb{D}$ (with $clust\_size = 2$), relying on the observation that the deviations in the gyroscope signal during PUPD events would be higher than those during writing events. Subsequently, we obtain two clusters Cl_1 and Cl_2. Since PUPDs are the occasional events, hence ideally, the cluster corresponding to the boardwork micro-gestures would be larger than that of the PUPD micro-gestures; we label the two clusters accordingly. However, there can be certain points with high deviations even within a single stroke (ex. while writing the English character 'K' – there is a sudden change in the axes orientation when one switches from the stroke '$\diagup$' to the stroke '$\diagdown$') – we call these points as the *sparse points*. These *sparse points* need to be separated from the PUPD micro-gestures.

We detect and eliminate the sparse points by applying a second level k-means clustering ($clust\_size = 2$) on the smaller cluster Cl_c (say, $c = argmin(|Cl_1|, |Cl_2|)$), obtained in the previous step. Let $\bar{Cl}_1$ and $\bar{Cl}_2$ be the resultant clusters, where $Cl_{\bar{c}}$ (say, $\bar{c} = argmin(|\bar{Cl}_1|, |\bar{Cl}_2|)$) be the smaller one. We compute the Silhouette index [20] σ and $\bar{\sigma}$ of the candidate clusters Cl_c and $Cl_{\bar{c}}$ respectively, which indicate the cohesivity of the corresponding clusters. Among the two candidate clusters Cl_c and $Cl_{\bar{c}}$, we choose the cluster with the highest silhouette index, which eventually reflects the PUPD micro-gestures. We designate this cluster as PUPD cluster and denote as Cl_{PUPD}. The other clusters represent the boardwork micro-gestures; hence it is expected that the deviations depicting strokes in character 'K' will be in other clusters. Once we get the cluster corresponding to the deviation points under the PUPD events, we drop the locus corresponding to the time instances (i, j) associated with each of the deviation points in that cluster.

V. Evaluation Methodology

To evaluate the accuracy of the generated transcripts, we need ground truth writing data with which the generated transcripts can be compared with. To collect such ground-truth data, we used either (a) a Wacom Pen-tablet, or (b) a touchscreen monitor, depending on the availability of the same with the volunteers, both mounted vertically on a wall.

A. Dataset Used for Experiments

The dataset used for the experiments are scripts from two languages - English and Bengali. For each of the languages, we consider all the characters and digits along with mathematical symbols such as ('+', '-', 'x', '/', '(', ')', '{', '}', '[', ']') for writing them individually over a whiteboard. Consequently, we have a total of 133 characters – 52 uppercase and lowercase alphabets with 10 digits in English, 51 alphabets with 10 digits in Bengali, and 10 mathematical symbols. Apart from the characters, we consider 36 unique English words[3] and 36 unique Bengali words. For the English words, we use two variants – (a) all characters in lower case, (b) only the first character in upper case, and rest in lower case; this gives us a total of 72 English words. The length of the words vary from 2 characters to 5 characters. The words have been chosen based on combinations of phonetics – like the corresponding vowels and consonants in both the datasets.

B. Human Experiments

We recruited 10 volunteers for the human study[4], including 8 male and 2 female, aged between 20-35 years. Out of the 10 participants, one of the participants used the left hand for writing. The participants scribe over the writing medium wearing a Moto360 smartwatch (with Android Wear OS 2.0) on the preferable wrist (left or right) in their convenient way. For capturing the ground-truth data and separating the boardwork activities from others, we instructed the participants to hold the pen/stylus at a fixed position for 2 seconds, before

[3] 'About','And', 'Apple', 'Been', 'Buy', 'Can', 'Cow', 'Day', 'Even', 'For', 'Game', 'Green', 'Happy', 'Have', 'Hello', 'Human', 'Issue', 'Job', 'Know', 'Long', 'More', 'New', 'One', 'Play', 'Quick', 'Run', 'Seven', 'Some', 'The', 'Under', 'Very', 'Was', 'World', 'Xenon', 'Year', 'Zone'.

[4] Institute Ethical Committee approval for human experiments is obtained. The volunteers have signed a consent form to participate in the data collection process.

and after each scribe, which indicates the start and the end of each writing session. In these set of experiments, the volunteers have not been given any training a priori, and they have been asked just to use the system freely.

C. Performance Metrics

Based on the collected ground truths, we evaluate the performance of *WatchScribe* from two different aspects – (a) visual similarity between the ground-truth scribbles and the generated transcripts, depicting the *decipherability* of the produced transcripts on the virtual whiteboard, and (b) metric-based evaluation to quantify the disparity between the ground-truth scribbles and the generated transcripts. The details follow.

1) Decipherability of the Generated Transcripts: To analyze the decipherability of the generated transcripts, we perform a human-driven evaluation, where we recruited 24 validators (different from the volunteers) to decipher the texts in the generated transcripts. We shared a form with the validators displaying images of generated transcripts. For each of the transcripts, the validators had three chances to decipher the corresponding text. From the validation inputs, we compute the i^{th} chance accuracy, termed as *i-accuracy* as follows. Let there be $\mathcal{N}$ different validators, and a transcript has been correctly recognized by $\mathcal{D}_i$ number of validators within i^{th} chances. Then i-accuracy ($\mathcal{A}_i$) for that transcript is defined as $\mathcal{A}_i = \frac{\mathcal{D}_i}{\mathcal{N}}$. We observed a marginal improvement (0.61%) in $\mathcal{A}_3$ compared to $\mathcal{A}_2$. However, the improvement is considerable (3.76%) if we move from $\mathcal{A}_1$ to $\mathcal{A}_2$. This indicates that the second chance is sufficient to decipher most of the transcripts. Accordingly, we use $\mathcal{A}_2$ in our evaluations and use the term *'accuracy'* to indicate the decipherability measured with $\mathcal{A}_2$.

2) Disparity with the Ground Truth: The key to designing the scheme is to compute if the two images of a board-work scribble, the ground-truth image and the image of the reconstructed transcript, have the same pixels highlighted, and if not, then how large is the disparity. Locality sensitive hashing can be used to compute such a disparity measure [21], [22]. We compute the disparity between the images α_{ws}, α_{gp} constructed from different methods – *WatchScribe* & *GyroPen* [16], respectively, and the ground truth α_g represented by the image of the ground truth script. The main steps are as follows. First, we crop all the images (generated & ground truth images) to eliminate the white border on all sides. Next, we resize all the cropped images to the same size $\tau \times \tau$ (here 150×150). Finally, we compute the disparity score ξ_{ig} between the resized generated image $\bar{\alpha}_i, i \in \{ws, gp\}$ and the resized ground truth image $\bar{\alpha}_g$. Fig. 5 illustrates the steps.

First, we partition both the images $\bar{\alpha}_i$ & $\bar{\alpha}_g$ into grids with grid size $\chi^j \times \chi^j$. We bitmap the grid partitioned image (say) $\bar{\alpha}_i$ by placing $\rho_i^j(k,l) = \{1, 0\}$ depending on the presence of the images inside each grid cell (k,l). Considering the cell (k,l) of two bitmapped images $\bar{\alpha}_i$ & $\bar{\alpha}_g$, we assign the cell-wise hamming distance $\gamma_{ig}^j(k,l) = 0$ if $\rho_i^j(k,l) = \rho_g^j(k,l)$, otherwise set $\gamma_{ig}^j(k,l) = 1$. Subsequently, we aggregate the

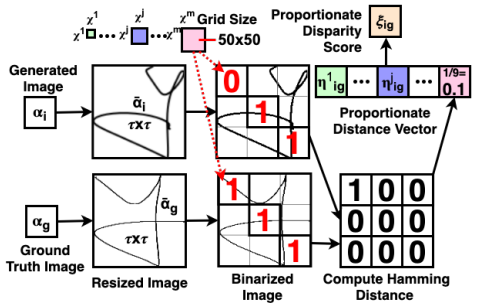


Fig. 5: Metric-based Framework for Quantifying Performance

distances of all the cells and compute the proportionate hamming distance score $\eta_{ig}^j = \sum_{h=1}^{\lambda^j} \sum_{l=1}^{\lambda^j} \gamma_{ig}^j(k,l)/(\lambda^j \times \lambda^j)$, where $\lambda^j \times \lambda^j$ is the cell count in the image partitioned with grid size $\chi^j \times \chi^j$. We repeat the procedure for m different grid sizes $\mathbb{X} = \{\chi^1 \times \chi^1, \ldots, \chi^j \times \chi^j, \ldots, \chi^m \times \chi^m\}$ and compute the proportionate hamming distance score for all the grids $\mathbb{X}$ as $\{\eta_{ig}^1, \ldots, \eta_{ig}^j, \ldots, \eta_{ig}^m\}$. Finally, eliminating the bias of the grid size, we compute proportionate disparity score between image $\bar{\alpha}_i$ & $\bar{\alpha}_g$ as $\xi_{ig} = \frac{1}{|\mathbb{X}|} \sum_{j=1}^{m} \eta_{ig}^j$. In our implementation, we consider $m = 6$, and the grid sizes are $10 \times 10, 15 \times 15, 20 \times 20, 30 \times 30, 40 \times 40, \&50 \times 50$.

VI. Performance Evaluation

If not otherwise mentioned, we consider maximum script-height and writing speed as $1.5''$ and 3.5sec per character, respectively, with the arm position at the writer's toe-to-shoulder height. Notably, we observed that the original handwriting (ground-truth) of an individual is always decipherable. Consequently, the accuracy measure correctly captures the decipherability of the generated transcripts.

Fig. 6: Ground Truth vs. *WatchScribe*-generated Words

A. Word Decipherability

Fig. 6 shows the visual impressions for the ground-truth scribbles and *WatchScribe*-generated transcripts of a few representative words from the English and the Bengali scripts. Table I gives a complete analysis of the decipherability of the *WatchScribe*-generated words. A *perfect match* indicates that the volunteer has completely deciphered the generated transcript (along with the case of the first character for English words). We also see the percentage of longest common subsequence (LCS) between the detected words and the ground-truth word, as indicated in *LCS match*. For example, if the word *'Xenon'* is detected as *'Venom'*, then LCS match is $3/5$. Additionally, we compute the accuracy by ignoring one (*Ignoring 1-char*) and two (*Ignoring 2-char*) characters while matching the detected word and the corresponding ground-truth. Finally, for English words, we compute the accuracy by

TABLE I: Decipherability of *WatchScribe*-generated Words ($\mathcal{L}$ and 'All' indicate the number of characters in a word and the overall performance across all words, respectively. Red numbers indicate the maximum accuracy within respective groups.)

Chance	*Perfect*				*LCS*				*Ignore-case (English Only)*			
	$\mathcal{L} \leq 3$	$\mathcal{L} = 4$	$\mathcal{L} \geq 5$	***All***	$\mathcal{L} \leq 3$	$\mathcal{L} = 4$	$\mathcal{L} \geq 5$	***All***	$\mathcal{L} \leq 3$	$\mathcal{L} = 4$	$\mathcal{L} \geq 5$	***All***
First	72.78	67.45	66.40	**69.37**	86.67	84.37	79.50	**83.93**	85.42	89.58	83.33	**86.11**
Second	**76.66**	70.83	67.20	**72.22**	**88.15**	85.68	79.82	**85.02**	86.46	**91.67**	84.38	**87.50**
Chance	***Ignoring-1-char***				***Ignoring-2-char***							
	$\mathcal{L} \leq 3$	$\mathcal{L} = 4$	$\mathcal{L} \geq 5$	***All***	$\mathcal{L} = 3$	$\mathcal{L} = 4$	$\mathcal{L} \geq 5$	***All***				
First	91.29	85.16	77.42	**85.50**	**95.46**	92.19	84.41	**90.80**				
Second	**91.85**	85.94	78.23	**86.19**	**95.46**	92.19	84.41	**90.80**				

TABLE II: Language-wise Decipherability of Words. Red numbers indicates maximum accuracy within respective groups.

Chance	*Perfect*		*LCS*		*Ignoring-1-char*		*Ignoring-2-char*		*Ignore-case*	
	Eng	***Ben***	***Eng***	***Ben***	***Eng***	***Ben***	***Eng***	***Ben***	***Eng***	***Ben***
First	69.79	68.52	87.29	77.21	90.28	75.93	**96.53**	73.61	86.11	-
Second	**73.61**	69.44	**88.70**	77.67	**91.32**	75.93	**96.53**	73.61	**87.50**	-

ignoring the case of the first character (*Ignoring case*). From the table, we observe that the perfect match accuracy is nearly 70%. Similar to the characters, we observe that the validators got confused where the first character looks similar in both the uppercase and the lowercase variations (e.g. *'Was'* and *'was'*). If we ignore the case for the English characters, we see that the accuracy improves up to $\sim 87\%$. LCS and ignoring one or two characters further improve the accuracy. Table II shows the decipherability of *WatchScribe*-generated words for the two languages. The decipherability of English words is better than Bengali words, primarily because the Bengali words are more cursive and hence opens more scopes for the confusion. To explore how the cursiveness in the writing impacts the performance of *WatchScribe*, we dig further with character decipherability, as discussed next.

B. Character-level Evaluation

We consider three types of English and Bengali alphanumeric characters – (a) single micro-gesture closed-curve (e.g. 'O'), (b) single micro-gesture open-curve (e.g. 'C'), and (c) multiple micro-gestures (e.g. 'Y'). Fig. 7 gives a visual comparison of the generated transcripts using *WatchScribe* and *GyroPen*, along with the corresponding ground-truth (image of the original handwriting captured from the stylus). Visually we observe that the generated transcripts for the open-curve single micro-gesture characters are very close to the ground-truth. For the closed curve single micro-gesture characters, although the generated transcripts are visually close to the ground-truth, the curve does not completely close in the generated transcript. As *WatchScribe* uses the angular displacement around the x-axis to estimate the locus of the pen over the yz-plane (Section III-C), even a small noise in the gyroscope reading significantly impacts this approximation. Such error during the locus generation gets visible for the closed-curve characters. However, as we can see even visually, the error is less in *WatchScribe* compared to *GyroPen* – fixing the reference frame correctly (Section III-B) helps us to reduce this error. Finally, we observe that the major improvement in *WatchScribe* in comparison with *GyroPen* comes for the characters having multiple micro-gestures (last row, Fig. 7). For such characters, the PUPD micro-gesture detection module helps us eliminate the locus of the pen, which is not a part of the original character. Therefore, the generated transcript with *WatchScribe* is closer to the ground-truth than the transcripts generated with *GyroPen*.

To delve more in-depth, we consider all the alphanumerics in the English alphabet set. Fig. 8 shows the average accuracy of individual characters across all the volunteers who participated in the data collection process. For this computation, we consider the decipherability instances from the majority ($2/3$rd) of the validators. We observe that the decipherability of some characters is very low. To further explore the reason behind the poor decipherability of these characters, we generate a heatmap, as shown in Fig. 9, where the rows correspond to the ground-truth character, and the columns correspond to the detected characters by the volunteers. Each entry (i, j) in the heatmap indicates the fraction of instances when the ground-truth character c_i has been confused with (detected as) another character c_j. The diagonal line indicates the complete decipherability. We observe four different scenarios where the decipherability is poor. (i) **Similar curves for upper and lower case letters:** (indicated through yellow circles in the heatmap) There are characters in the English alphabet set, where the curves are similar for an uppercase and the corresponding lowercase letters (e.g. 'O'/'o', 'X'/'x'). The validators made mistakes in deciphering these characters. We do not see this as a direct limitation of *WatchScribe*, and evaluate this further with the metric-based evaluation, as discussed later. (ii) **Similar cursive scripts:** (purple circle in the heatmap): Some characters (e.g. 'k' and 'R') look similar on cursive writing. As we have seen earlier, since *WatchScribe* uses angular displacement to estimate the locus, it makes the scripts a bit more cursive due to error introduced from gyroscope measurement noise. Therefore, the decipherability of these characters gets reduced. Nevertheless, the decipherability level depends on the writing style. (iii) **Issue with vertical lines:** (the black circle in the heatmap): Because of the error introduced during angular displacement estimation, the vertical lines produced by *WatchScribe* becomes a bit cursive. Consequently, it reduces the decipherability of cer-

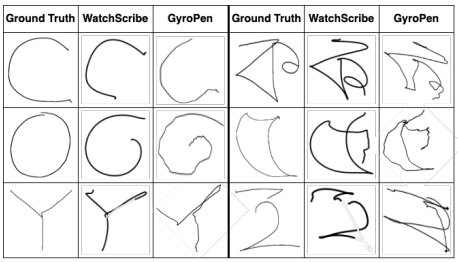


Fig. 7: Generated Scribbles on alphanumeric characters

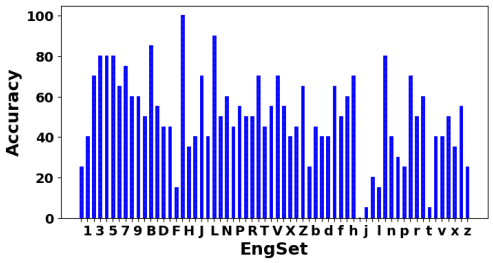


Fig. 8: Performance of *WatchScribe* on all English alphanumeric characters

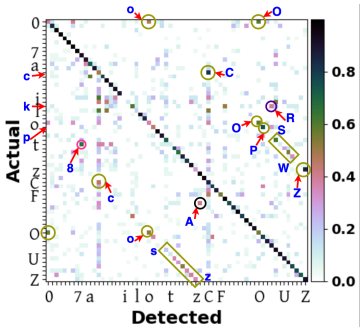
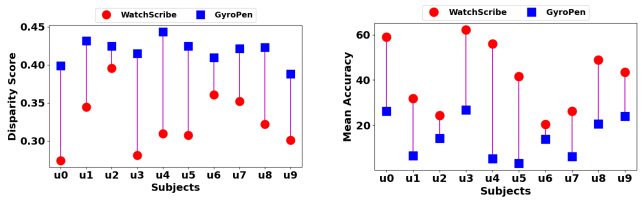


Fig. 9: Individual Transcription Performance

tain characters where the vertical lines' alignment has minor differences (e.g. 'A' and 'H'). (iv) **Issue with PUPD micro-gestures:** (pink circle in the heatmap) As *WatchScribe* uses a complete unsupervised approach for PUPD micro-gesture detection, sometimes we observe that the proposed method can not completely eliminate the unnecessary micro-gestures. Consequently, we see a slight curve at the beginning and/or at the end of some strokes for a multi-stroke character, which creates confusion among a few characters (e.g. 't', 'b', '8').

In summary, we observe that the *median* accuracy of character transcripts generated by *WatchScribe* is 66.67% for both English and Bengali scripts. In contrast, the *mean* accuracy is 64.52% for English and 56.28% for Bengali, computed over all the characters from both scripts. The interesting observation from this analysis is that the accuracy of word-decipherability is more than that of the character-decipherability. As discussed in [23], the graphic design techniques differentiate **legibility** (ease with which *a symbol* can be decoded) from **readability** (ease with which a reader can understand *words or sentences*). Interestingly, our evaluation of decipherability checks the legibility of individual characters generated by *WatchScribe*. On the other hand, it measures the readability of the *WatchScribe*-generated words. While legibility requires a near-accurate representation of the symbols, readability inherently uses the dictionary of words based on the language's proficiency [24]. Therefore, a user might be able to decipher the complete word even if she has not been able to decipher all the characters of that word. Consequently, we observe that a ∼60% accuracy in the character-decipherability can help achieve ∼70% accuracy in the word-decipherability.



(a) Disparity (b) Decipherability

Fig. 10: Performance across Subjects

C. Performance Comparison with Baselines

Next, we proceed to evaluate how *WatchScribe* fares in terms of both the decipherability and the disparity score, compared to other baselines. Fig. 10a and Fig. 10b show the overall disparity scores and the mean accuracy for decipherability, respectively, across all the characters (over both English and Bengali scripts) for individual subjects. We observe that the disparity score with *WatchScribe* is ∼10% less than *GyroPen*, on average. For some subjects (e.g. u1, u2, u6, & u7), the disparity score is higher than others. A close inspection shows that these subjects mostly uses block scripts for handwriting, where the usage of straight lines is more prominent than cursive scripts. As we have seen earlier, the straight lines become a bit cursive at the transcriptions generated by *WatchScribe*. Intuitively, the disparity between a straight line and the corresponding cursive line is high based on the metric designed in Section V-C2. Further, as we have seen earlier, the cursive characters are the sources of confusion, so the mean accuracy for decipherability gets reduced for these users.

VII. Conclusion

To the best of our knowledge, *WatchScribe* is the first of its kind, which uses smartwatch data to reproduce real-time on-device transcription of boardworks with zero bootstrapping. From thorough testing over ten subjects, *WatchScribe* shows promising results over a large pool of test cases, including 133 characters in total and 36 unique words each from the English & Bengali languages. Compared to the existing approaches on reproducing handwriting through smart devices, our method uses a complete unsupervised approach and thus requires zero bootstrapping. During the development of *WatchScribe*, we observed that *WatchScribe* makes the straight lines a bit cursive due to the error introduced during the estimation of the angular displacement. The performance of *WatchScribe* can be improved further if this error can be reduced. As a next step, we target to incorporate these extensions over *WatchScribe* so that even very complex, unorganized, and random scribbles can be replayed over a virtual whiteboard with the help of a smartwatch.

REFERENCES

[1] A. M. Reports, "Global smartwatch market size, market share, application analysis, regional outlook, growth trends, key players, competitive strategies and forecasts, 2018 to 2026," January 2019.

[2] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and I can track my user's arm," in *Proceedings of the* 14^{th} *ACM international conference on Mobile systems, applications, and services*, 2016, pp. 85–96.

[3] C. Bi and G. Xing, "Real-time attitude and motion tracking for mobile device in moving vehicle," in *Proceedings of the* 16^{th} *ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 357–358.

[4] H. Zhou, Y. Gao, X. Song, W. Liu, W. Dong, and Y. Jiang, "Wearable-based human-computer interaction with limbmotion," in *Proceedings of the* 16^{th} *ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 339–340.

[5] C. Bi, J. Huang, G. Xing, L. Jiang, X. Liu, and M. Chen, "SafeWatch: a wearable hand motion tracking system for improving driving safety," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 1, pp. 1–21, 2019.

[6] G. Reyes, J. Wu, N. Juneja, M. Goldshtein, W. K. Edwards, G. D. Abowd, and T. Starner, "SynchroWatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–26, 2018.

[7] 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* 16^{th} *International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 9–14.

[8] L. Ardüser, P. Bissig, P. Brandes, and R. Wattenhofer, "Recognizing text using motion data from a smartwatch," in *Proceedings of IEEE International Conference on Pervasive Computing and Communication Workshops*, 2016, pp. 1–6.

[9] X. Lin, Y. Chen, X.-W. Chang, X. Liu, and X. Wang, "SHOW: Smart handwriting on watches," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, p. 151, 2018.

[10] R. Wijewickrama, A. Maiti, and M. Jadliwala, "deWristified: handwriting inference using wrist-based motion sensors revisited," in *Proceedings of the* 12^{th} *Conference on Security and Privacy in Wireless and Mobile Networks*, 2019, pp. 49–59.

[11] H. Jiang, "Motion Eavesdropper: Smartwatch-based handwriting recognition using deep learning," in *Proceedings of ACM International Conference on Multimodal Interaction*, 2019, pp. 145–153.

[12] Q. Xia, F. Hong, Y. Feng, and Z. Guo, "MotionHacker: Motion sensor based eavesdropping on handwriting via smartwatch," in *IEEE INFOCOM Workshops*, 2018, pp. 468–473.

[13] Y. Yin, L. Xie, T. Gu, Y. Lu, and S. Lu, "AirContour: Building contour-based model for in-air writing gesture recognition," *ACM Transactions on Sensor Networks*, vol. 15, no. 4, p. 44, 2019.

[14] Q. Dai, J. Hou, P. Yang, X. Li, F. Wang, and X. Zhang, "The sound of silence: end-to-end sign language recognition using smartwatch," in *Proceedings of the* 23^{rd} *Annual International Conference on Mobile Computing and Networking*, 2017, pp. 462–464.

[15] J. Hou, X.-Y. Li, P. Zhu, Z. Wang, Y. Wang, J. Qian, and P. Yang, "Signspeaker: A real-time, high-precision smartwatch-based sign language translator," in *Proceedings of the* 25^{th} *International Conference on Mobile Computing and Networking*, 2019, pp. 1–15.

[16] T. Deselaers, D. Keysers, J. Hosang, and H. A. Rowley, "Gyropen: Gyroscopes for pen-input with mobile phones," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 263–271, 2014.

[17] M. Pedley, "Tilt sensing using a three-axis accelerometer," *Freescale semiconductor application note*, vol. 1, pp. 2012–2013, 2013.

[18] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter, "Using mobile phones to write in air," in *Proceedings of the* 9^{th} *international conference on Mobile systems, applications, and services*, 2011, pp. 15–28.

[19] G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications," *IEEE transactions on robotics and automation*, vol. 17, no. 5, pp. 731–747, 2001.

[20] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[21] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the* 30^{th} *ACM symposium on Theory of computing*, 1998, pp. 604–613.

[22] M. Astefanoaei, P. Cesaretti, P. Katsikouli, M. Goswami, and R. Sarkar, "Multi-resolution sketches and locality sensitive hashing for fast trajectory processing," in *Proceedings of the* 26^{th} *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018, pp. 279–288.

[23] J. E. Sheedy, M. V. Subbaram, A. B. Zimmerman, and J. R. Hayes, "Text legibility and the letter superiority effect," *Human factors*, vol. 47, no. 4, pp. 797–815, 2005.

[24] E. Qiao, F. Vinckier, M. Szwed, L. Naccache, R. Valabrègue, S. Dehaene, and L. Cohen, "Unconsciously deciphering handwriting: subliminal invariance for handwritten words in the visual word form area," *Neuroimage*, vol. 49, no. 2, pp. 1786–1799, 2010.