Using Mobile Phones to Write in Air

Sandip Agrawal
Department of ECE
Duke University
Durham, NC, USA

Ionut Constandache
Department of CS
Duke University
Durham, NC, USA

Shravan Gaonkar
Department of CS
University of Illinois
Durham, NC, USA

Romit Roy Choudhury
Department of ECE
Duke University
Durham, NC, USA

Kevin Caves
Speech Pathology & Audiology
Duke Medical School
Durham, NC, USA

Frank DeRuyter
Speech Pathology & Audiology
Duke Medical School
Durham, NC, USA

ABSTRACT

Numerous sensors in modern mobile phones enable a range of people-centric applications. This paper envisions a system called *PhonePoint Pen* that uses the in-built accelerometer in mobile phones to recognize human writing. By holding the phone like a pen, a user should be able to write short messages or draw simple diagrams in the air. The acceleration due to hand gestures can be translated into geometric strokes, and recognized as characters. We prototype the *PhonePoint Pen* on the Nokia N95 platform, and evaluate it through real users. Results show that English characters can be identified with an average accuracy of 91.9%, if the users conform to a few reasonable constraints. Future work is focused on refining the prototype, with the goal of offering a new user-experience that complements keyboards and touch-screens.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Interaction Styles, Input Devices and Strategies*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*User Interfaces*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human Factors*

General Terms

Algorithms, Design, Experimentation, Human Factors

Keywords

Gestures, Activity Recognition, Accelerometers, Smartphones

1. INTRODUCTION

Imagine the following scenario. While driving to office, Bob stops at a traffic light. As he mentally sifts through his tasks for the day, he remembers that he needs to call his friend, Alice, very soon. Since Bob tends to forget his personal commitments, he decides to make a note of this task. Therefore, while keeping his gaze on the traffic lights, he draws out the phone from his pocket, and by holding it like a pen, he writes "ALICE" in the air. He also gestures a check-mark to email the written note to himself. He does not look at any of the hand-gestures he makes. Once in his office, he finds an email in his mailbox that reads "PhonePoint Pen – ALICE". Bob calls Alice, talks to her, and deletes the email. The figure below shows the output of air-writing ALICE using the PhonePoint Pen.

The above is a fictional scenario, however, representative of broad possibilities in the area of human-sensor interaction. A particular possibility pertains to a sensor-assisted input technology that can easily "note down" short pieces of information. Although existing technologies have made valuable advances to meet these needs, the quality of user-experience could still improve. We discuss some avenues of improvement, and motivate the potential of PhonePoint Pens.

Typing an SMS, while popular among the youth, has been unpopular among a moderate section of society. Studies report user dissatisfaction with mobile phone typing [24, 7, 6]. The major sources of discomfort arise from small key sizes, short inter-key spacings, and the need for multi-tapping in some phone keyboards. The difficulties are likely to be pronounced for elderly people and motor impaired patients.

Even if future keyboards [22] improve the typing experience, some problems may still persist. While walking, or with one hand occupied, typing in information may be inconvenient. Using the mobile phone accelerometer to capture hand gestures, and carefully laying them out in text or image, can improve the user experience. The ability to write without having to look at the phone keypad may offer an added advantage.

One may argue that voice recorder applications on mobile phones may be an easy way to input short pieces of information. However, searching and editing voice-recorded content is difficult (unless processed through a separate speech-to-

MobiSys'11, June 28–July 1, 2011, Bethesda, Maryland, USA.

text software). Further, playing aloud the voice messages can be inconvenient and time-consuming. Writing in air, and converting them to typed text, may alleviate these problems.

Current approaches are largely ad hoc. People use whatever is quickly reachable, including pen-and-paper, sticky notes, one's own palm, etc. None of these scale because they are not always handy, and more importantly, not always connected to the Internet. Thus, hastily noted information gets scattered, making information organization and retrieval hard.

This paper proposes to use the in-built accelerometer in modern mobile phones as an easy and ubiquitous way of capturing (short) written information. The problem definition bears similarity to known problems in gesture recognition. However, as we will see later, recognizing actual alphabets in air (using the phone processor, a noisy accelerometer, and no software training), raises a number of new challenges. For instance, as a part of writing the alphabet "A" on paper, one must write "/\" first, lift and reposition the pen on the paper, and then write the "—". When writing in air, the phone cannot easily say which part of the hand-movement is intended to be the "re-positioning" of the pen. The problem is further complicated by the inherent noise in mobile phone accelerometers, the user's involuntary wrist-rotation, and practical difficulties in deriving displacement from noisy acceleration. Simplicity is also essential to ensure that all the operations can be performed on the phone processor. The PhonePoint Pen (P3) attempts to address these challenges by treating the accelerometer readings as a digital signal, and successively refining it through simple numerical and signal processing algorithms. Once individual geometric movements have been tracked, their sequence of occurrence is matched against a decision tree (a simple grammar). The matching operation yields the English alphabet, which are then juxtaposed to form words.

Our current prototype is not yet ready for public use – it is an early step towards a difficult problem, and some deficiencies remain unaddressed. Mainly, we advised users to pretend that they are writing on an imaginary blackboard while holding the phone like an blackboard eraser – this reduces the user's wrist and elbow rotation while gesturing a stroke. Users were also requested to write 12-inch sized capital letters at a moderate speed (not too fast), to ensure that the sampling frequency of the accelerometer was adequate to capture the hand-motions. Lastly, users were asked to briefly pause between strokes while writing letters – this allowed P3 to correct inaccuracies in phone displacements caused by the inherent noise in accelerometer readings. Users who performed the tests after 3 to 4 minutes of rehearsing, achieved an average accuracy of 91.9% with English alphabets. The geometric representation of the characters (shown as 2D images of the actual hand-writing) were legible in around 83% of the cases. Survey responses from randomly picked student users, as well as from real patients and doctors in the Duke University Hospital, were positive. The absence of visual feedback while writing did not appear to be a concern at all, and the energy consumption from air-writing was marginal. While further research is certainly necessary to attain a natural and intuitive input system, we believe that the P3 prototype presents promise of viability in the near future. Our main contributions may be summarized as follows.

- **We explore the viability of using the mobile phone accelerometer to write in the air.** While a number of gesture recognition schemes already exist [8, 9, 3, 14], the ability to write English alphabets (and draw simple diagrams) present distinct challenges.

- **We characterize the nature of the challenges and propose a multi-phase approach to recognize alphabets and words.** Our algorithms are deliberately simple for on-phone real-time operation. We also develop a customized spell checker that corrects motion-related errors, such as between D and P.

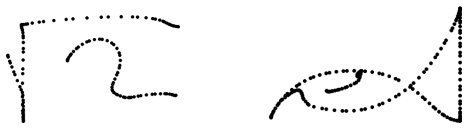
- **We prototype the PhonePoint Pen (P3) on Nokia N95s, and test it with** 10 **student users and** 5 **hospital patients.** All but one student user were able to write with good accuracy, and their geometric outputs were quite legible. Accuracy with real patients was poor (partly due to usability problems), however, the feedback from medical practitioners were unanimously encouraging.

The rest of the paper expands on these contributions. We begin with a discussion of possible use cases of P3.

2. USE CASES

(1) Assistive Communications for Impaired Patients:
The Speech Pathology and Surgery division of Duke University Medical School expressed keen interest in using a future, more refined prototype of the PhonePoint Pen to serve as an assistive technology for impaired patients. Several patients suffer from inherent speech impairments, or experience similar conditions after surgeries. War veterans often loose fingers or limbs, while others lack finger-dexterity for typing on keypads. Yet, these patients are often capable of broad (one-handed) gestures, such as in sign-languages. A mature version of P3 can be of assistance to such patients. It can permit some level of impromptu communication between a speech/hearing-impaired patient and someone who does not understand sign-languages. We have performed 15-minute experiments with 5 patients at the Duke University Hospital, and discussed the applicability of the system with surgeons and healthcare professionals. We discuss the experience in Section 5.4.

(2) Equations and Sketching:
One of the P3 test users suggested the possibility of quickly writing equations in the air. Equations are difficult to write with regular phone keyboards, and P3 may be convenient. Other possible use-cases involve taking mental notes, sketching simple diagrams and driving directions, or drawing a desired food item (e.g., fish) in a foreign country's restaurant. At present, P3 is unable to draw figures with high reliability – the following fish and equation were drawn in 2 attempts each.



(3) Emergency Operations and First Responders: Emergency scenarios are often unsuitable for typing, or even talking on the phone, because the observer may be engaged in looking at the events around her. P3 allows taking notes (with one hand), and more importantly, without requiring visual attention. The ability to look around and gesture at the same time may be useful in these critical situations.

While the above use-cases are specific to phone-based systems, one may imagine more creative applications of air-writing (e.g., writing "CNN" in the air may switch the TV to the CNN news channel). We envisioned PhonePoint Pen with the aim of enabling a wide range of such people-centric applications. While the current P3 prototype falls short of realizing all these applications, we believe that P3 establishes feasibility, and justifies longer-term research commitment.

3. CORE CHALLENGES

Existing systems, such as the Wii[20], PlayStation Move [23], Xbox Kinect [30], and others [31, 32, 9, 3, 14], have the ability to identify hand gestures with good accuracy. These gestures have been utilized to recognize a few numeric digits [31, 32]. Moreover, several of these systems are more resourceful in sensor hardware, including gyroscopes, webcams, and more CPU power [2, 16, 5]. Writing English alphabets/words in real-time with commodity phones has been an unexplored problem. To this end, we discuss the main research challenges in P3, and present our initial approaches. We then assemble these building blocks into a functional prototype.

(1) Filtering Rotation without Gyroscope

Issue: Nokia N95 phones are equipped with a 3-axis accelerometer that detects acceleration in the X, Y, and Z directions. Figure 2(b) shows an example of raw accelerometer readings on each of the 3 axes. The accelerometers measure linear movement along each axis, but cannot detect rotation. Hence, if the human grip rotates while writing, the reference frame of acceleration gets affected. Existing devices like "Wii Motion Plus" and Airmouse employ a gyroscope to discriminate rotation [21, 16]. In the absence of a gyroscope, compensating for hand rotation is a problem[1].

Approach: We begin with a brief functional explanation of the gyroscope. Consider the position of a gyroscope-enabled phone (GEP) at time $t = t_0$ in 2D space (shown in the left side of Figure 1). At this initial position, the figure shows that the GEP's axes are aligned with the earth's reference axes (i.e., gravity is exactly in the negative Y direction). The accelerometer reading at this position is $< I_x(t_0), I_y(t_0) - g >$, where $I_x(t_0)$ and $I_y(t_0)$ are the instantaneous accelerations along the x and y axes at time t_0 respectively, and g is gravity. Now, the phone may rotate at the same physical position at time t_1 also shown in Figure 1 (right). The phone now makes an angle θ with the earth's reference frame, and the accelerometer readings are $< I_x(t_1) - gsin(\theta), I_y(t_1) - gcos(\theta) >$. However, it is possible that the phone moved along the XY plane in a manner that induced the same acceleration as caused by the rotation. This leads to an ambiguity that gyroscopes and accelerometers can together resolve (using angular velocity detection in gyroscopes). However, based on the accelerometer readings alone, linear movements and rotation cannot be easily discriminated.

[1]The new iPhone4 has a gyroscope, however, the phone was not released at the time of this work.

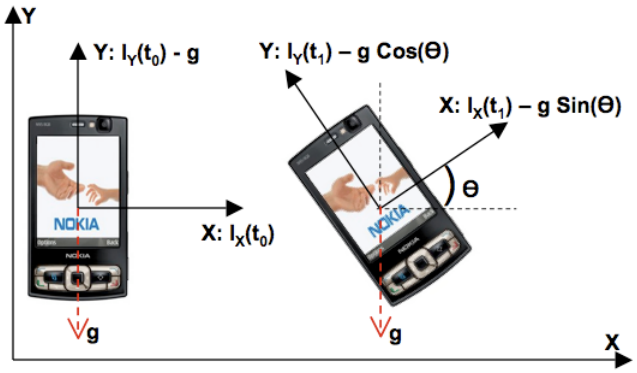


Figure 1: Earth's gravity projected on the XY axes; the axes are a function of the phone's orientation.

This is a difficult problem which we address by imposing a soft constraint on the user. We asked the user to pretend that one of the corners of the phone is the pen tip, and to hold it in a non-rotating grip (shown in Figure 2(a)). Some users also found it easier to hold it like a blackboard eraser – this grip also reduces wrist-rotation. In addition, while writing an alphabet, users were advised to briefly pause between two "strokes". The pause is often natural because the user changes the direction of movement (from one stroke to another). For example, while writing an "A", a pause after stroke "/" and before the starting of stroke "\" can be exploited. An accelerometer snapshot at this paused instance can identify the components of gravity on each axes, and hence, the angular orientation θ can be determined. Knowing θ, the phone's subsequent movement can be derived. Of course, we assume that the phone rotates only in-between two strokes, and not within any given stroke. If this assumption gets violated, P3's character recognition accuracy gets affected.

(2) Suppressing Background Vibration

Issue: Accelerometers are sensitive to small vibrations. Figure 2(b) reports acceleration readings as the user draws a rectangle using 4 strokes (around 350 units on the Z-axis is due to earth's gravity). A significant amount of jitter is caused by natural hand vibrations. Furthermore, the accelerometer itself has measurement errors. It is necessary to suppress this background vibration (noise) to extract jitter-free pen gestures.

Approach: To cope with vibrational noise, we apply two noise-reduction steps (the acceleration is treated as a discrete sequence of signal samples). First, we smooth the accelerometer readings by applying a moving average over the last n readings (in our current prototype, n=7). The results are presented in Figure 2(c). Next, we label all acceleration samples less than $0.5m/s^2$ as noise. We chose this threshold based on the average vibration caused when the phone was held stationary. All noise-labeled samples are suppressed (i.e., set to 0). Figure 3(a) shows the combined effect of smoothing and noise suppression.

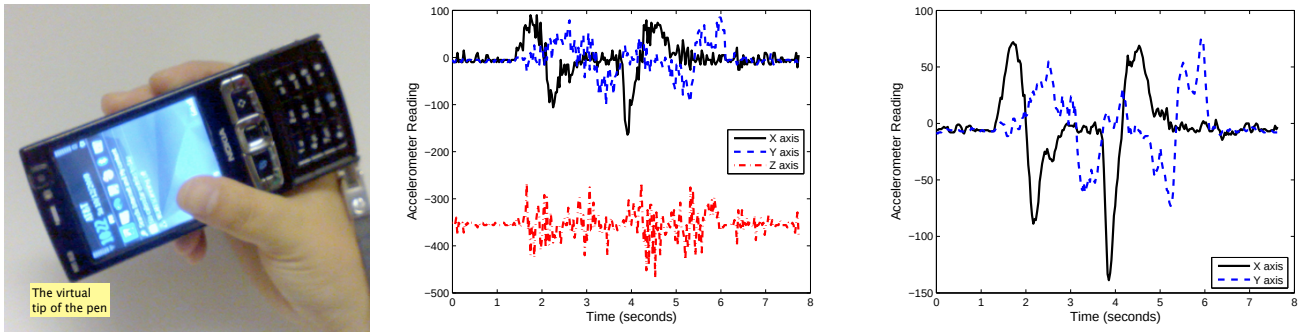


Figure 2: (a) Pretending the phone's corner to be the pen-tip reduces rotation. (b) Raw accelerometer data while drawing a rectangle (note gravity on the Z axis). (c) Moving average computation.

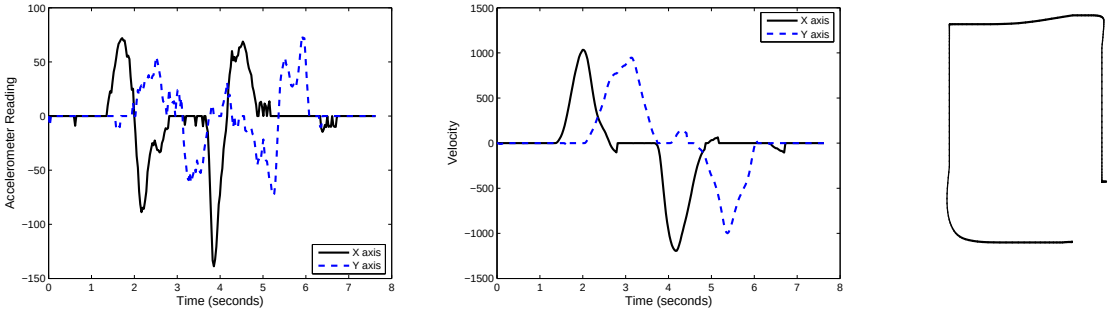


Figure 3: (a) Final processed acceleration readings (b)Computing velocity as an intermediate step towards measuring displacement. (c) The approximate rectangle as the final output.

(3) Computing Displacement of Phone

Issue: The phone's physical displacement is necessary to estimate the size of the air-written character, as well as their relative positions (such as in equations, figures, etc.). The displacement δ is computed as $\delta = \int \left(\int a \, dt \right) dt$, where a is the instantaneous acceleration. In other words, the algorithm first computes the velocity (the integration of acceleration), followed by the displacement (the integration of velocity). Noise in the acceleration readings will reflect on the velocity computation, and will get magnified in the computation of displacement. For instance, an erroneous short positive impulse in the accelerometer (i.e., acceleration becoming positive and then returning to zero), results in a positive velocity. Unless an identical negative impulse compensates for the positive impulse, the phone would appear to be in a state of continuous velocity. When this velocity is integrated, the displacement error will grow larger.

Approach: In order to reduce the velocity-drift errors, we look at consecutive accelerometer readings labeled as noise in the previous step. We reset the velocity to zero, if n consecutive readings have been suppressed as vibrational noise. This is because a continuous sequence of noise vibration is a good indicator of a pause, or a statically held phone; hence, it is an opportunity to suppress inertial error. Figure 3(b) shows the effect of resetting the velocity. Even if small velocity drifts are still present, they have a tolerable impact on the displacement of the phone. As seen in Figure 3(c) the amount of displacement and the shape drawn are represented reasonably well. The direction of movement is inferred from the signs of the acceleration along the X, Y, and Z axes.

(4) Differentiating an "A" from a Triangle

Issue: The imaginary blackboard on which the user air-writes has no visible reference frame for position. As a result, some characters become more difficult to write. To illustrate this source of confusion, let us consider a simple example. Assume the user is writing the character "A". The writer has already drawn the "/" and "\", and now lifts the pen to draw the "–". Observe that the phone has no idea about the global position of "/\". Hence, upon drawing the "–", the pen does not know whether it is meant to be added in the center (to indicate an "A"), or at the bottom (to indicate a triangle, Δ). This ambiguity underlies several other characters and shapes.

Approach: To address the above ambiguity, we jointly exploit the accelerations along the X, Y, and Z axes. While writing an "A", assume that the user has just finished writing "/\". The pen is now at the bottom of the "\". The user will now lift the pen and move it towards the upper-left direction, so that she can write the "–". The lifting of the pen happens in 3D space, and generates an identifiable impulse on the Z axis. When the acceleration in Z axis is above a certain threshold, we label that stroke as the "lifting of the pen". This pen-lifting is used as a trigger for the user going off the record. User movements in the XY plane are still monitored for pen repositioning, but are not included in the final output. When the phone is in position to write "–", a pause and change in direction is an indication for going back on the record.

(5) Identifying Character Transitions

Issue: Even if pen-lifts are recognized, certain ambiguities remain. For instance, "B" and "13" may have the exact same hand-movement, including the pen-lift. The user's intention is difficult to recognize, making character distinction hard.

Approach: We rely on a combination of multiple heuristics to mark character separations. The simplest approach is to require the user to include a special gesture between characters, like a "dot" or a relatively longer pause. Thus, "13" should be written as $|.\ \supset\supset$, while the "B" should be $|\ \supset\supset$. These delimiters are inspired from the Scriboli system [12] which employs distinct stylus-based gestures to select objects displayed on a Tablet PC. While these special gestures may be inconvenient, they provide a clear indication of user intent. In addition, P3 employs other methods for delimiting characters. These methods rely on understanding what the user has written till now, and what the next "stroke" is likely to be. We will discuss this in detail in the next section, after we have discussed stroke-detection and a simple stroke-grammar to identify characters.

4. SYSTEM DESIGN AND ALGORITHMS

The above building blocks provide for a geometric representation of air-written characters. While the geometric version can be displayed or emailed as an image, conversion to text is likely to be more useful (for browsing and searching). This section develops the algorithmic components towards this goal.

4.1 Stroke Detection

Characters can be viewed as a sequence of strokes. The alphabet "A", for instance, is composed of 3 strokes, namely "/", "\", and "—". If the discrete strokes can be pulled out from the seemingly continuous movement of the hand, it is possible to infer the characters. To this end, we have analyzed the English alphabets and constructed a basic set of strokes, as in Figure 4.

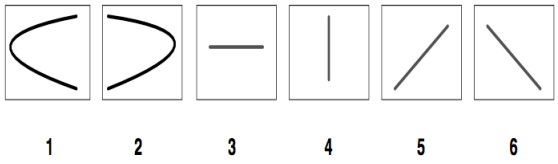


Figure 4: Basic strokes for English characters.

To identify the strokes, P3 computes a running variance of the accelerometer readings. When this variance falls below a threshold, P3 marks those regions as a pause of the hand. The pauses demarcate the human-strokes, allowing P3 to operate on each of them individually. For determining the exact stroke, our basic idea is to correlate the human-strokes against each of the ideal strokes. This form of correlation is not new, and has been used as standard primitives in classification and matching techniques [14, 8, 1]. We perform correlation over a varying window size of accelerometer readings. This is because the hand often rotates towards the end of the stroke, and the samples corresponding to the rotations should ideally be pruned out. Correlation is able to cope with such issues, showing a high correlation value when the ideal stroke *broadly aligns* with actual readings. Besides, even if some inter-stroke pauses are not identified, varying the correlation window-size yields the stroke boundaries. The intuition is that two consecutive strokes are typically different in the English alphabet, and thereby, correlating across the boundaries of the strokes (with a large window size) reduces the correlation value. Performance results indicate a reasonable reliability in stroke detection. The natural question, then, pertains to combining the strokes into a character.

4.2 Character Recognition

The PhonePoint Pen observes the logical juxtaposition of strokes to deduce the character that the human is trying to write. For this, we adopt a stroke grammar for English alphabets and digits. Figure 5 shows a pruned down version of this grammar for visual clarity. The grammar is essentially a tree, and expresses the valid sequence of strokes to form an alphabet. Moreover, the grammar also helps in stroke-recognition because it provides P3 with an ability to anticipate the next stroke. For instance, observing strokes "$|\ \backslash\ /$" in succession, P3 can anticipate an "M" and expect the next stroke to be a "|". Thus, by correlating "|" to the stream of accelerometer readings (and ensuring a high correlation), the system can better identify the end-points of the next stroke. This helps in identifying the residual samples, which in turn helps in tracking the re-positioning of the hand in-between strokes.

In certain cases, the user's hand movement may be falsely classified as an incorrect stroke. A frequent example is "\" and "$\supset$". Since the user's hand has a natural rotational motion (pivoted at the elbow), moving diagonally for a "\" results in an arc, which then gets classified as "$\supset$". Thus "N" may not be recognizable due to misclassification of the second stroke. To account for such possibilities, we have updated the grammar tree. For example, if | is followed by $\supset$, we call it a "D" or "P"; however, if this is again followed by a "|", we infer an "N" (since no alphabet is a sequence of "$|\ \supset\ |$"). We observe that such opportunities are numerous in the stroke grammar, adding to the robustness of the system. We do not include this updated grammar in the paper and only show the example for N in Figure 6.

Grammar Ambiguity. Interestingly, the stroke grammar presents a number of ambiguities. For instance, "O" and "S" are composed of the same strokes, namely, "$\subset$" and "$\supset$". P3 resolves this by simply observing the direction of movement in the second stroke. If the hand is moving upwards, computed from the sign of the Y-axis acceleration, the alphabet is declared as an "O", and the vice versa. Another ambiguity is between "D" and "P". In this case, P3 computes the relative sizes of Y-axis movements and compares them. If the sizes are comparable (second stroke greater than 0.75 of the first), the alphabet is deemed as a "D", else a "P". Finally, some kind of ambiguities are relatively harder. For instance, "X" and "Y" have the same strokes, and only differ in how the user repositions her pen. Since hand-repositioning has no preset movement, they are more prone to error. Thus, even though the "\" in "Y" is smaller than that of "X", P3 makes a mistake in some cases. Finally, "O" and "0" cannot be discriminated.

4.3 Word Recognition

Recognizing the juxtaposition of characters, to ultimately recognize a word, adds to the ambiguity. For instance, "B" and

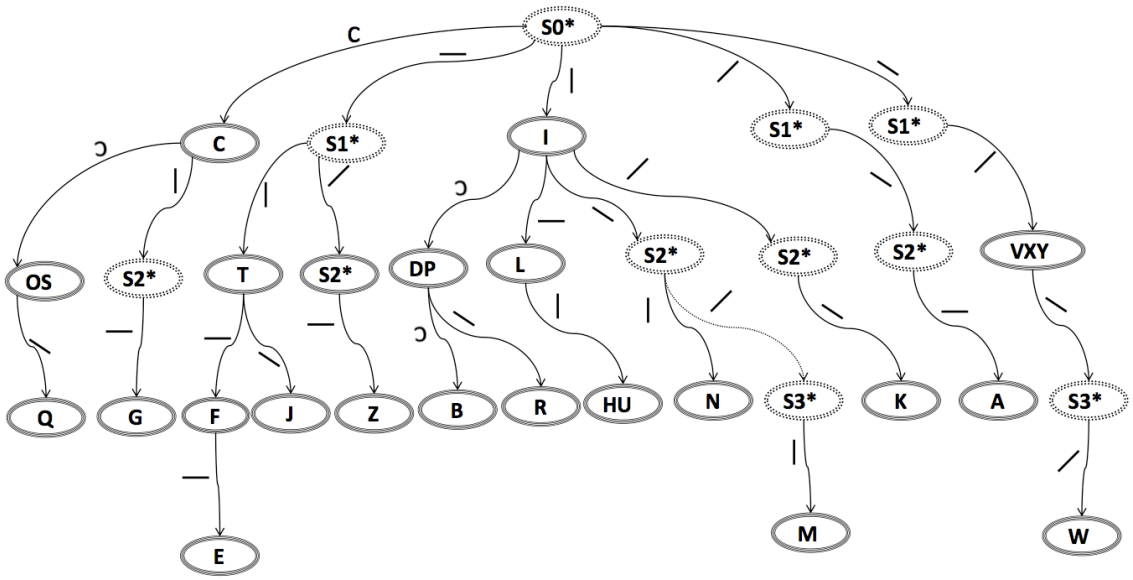


Figure 5: The basic grammar for character recognition. The edges are labeled with a stroke. Reaching certain states (solid circles) implies a valid alphabet, while others are intermediate states (dotted circles). Certain states have multiple alphabets in them, suggesting ambiguities in the basic stroke grammar.

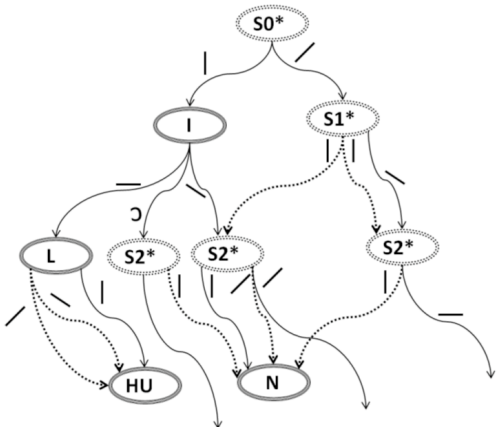


Figure 6: Incorporating tolerance into the grammar tree translating it into a graph. The dotted edges are incorrect but permissible, e.g., alphabet "N" can be reached via multiple paths such as | ⊃ |.

"13" are identical in terms of the strokes used, and so are "H" and "IT". Unless we find a signature to demarcate characters, the PhonePoint Pen will yield false positives. Towards this goal, we consider a combination of multiple heuristics. None of these heuristics are adequate in isolation, but may be reasonable when used in conjunction. First, we make the observation that while transitioning from one alphabet to the next, users have a naturally longer pause (especially with upper case alphabets). Second, we observe that in some cases, if the hand moves in a *leftward* direction, it may be a hint about the start of a new character. This happens, for example, when a user has written across the imaginary plane in front of her, and moves back in space (towards left) to write more. Typically, since most of English alphabet strokes are gestured left to right, any opposite movement (from right to left), is a useful hint for character segregation. Third, we ask the users to gesture a "dot" between characters whenever they can remember. Thus, "13" should be written as "|. ⊃⊃", while the "B" should be "| ⊃⊃". Drawing a dot presents a unique signature to delimit characters, but slows down the user while writing. Thus, the user can use it only if she remembers or wishes to. If the delimiter is not used, the recognition accuracy is affected.

We note that not all cases are like "B" and "13". Even without the delimiter, the stroke grammar will naturally separate some characters. In other words, given a sequence of strokes, P3 anticipates the next stroke to be from a specific subset of strokes. If the next stroke is not in this anticipated subset, then it implies the start of a new character. For example, given "|" and "—", the phone can anticipate a "—" assuming that the user is trying to write an "F", or "E". However, if the next stroke is "⊂", then the phone immediately infers that the prior alphabet was an intended "T". Even if the delimiter is not present, such character transitions can be recognized to form words. Spell checkers can be employed on top of these methods to further improve word recognition accuracy.

4.4 P3-Aware Spelling Correction

Spelling correction tools accept a given word and compute a list of valid English words, sorted in the order of "edit distance". The edit distance between two strings of characters is defined as the number of operations required to transform one of them into the other. The corrected spelling is typically the valid word with minimum edit distance. Importantly, multiple words may have the same (minimum) edit distance, and even the minimum edit distance may not be the best when the nature of the errors are guided by certain distributions. For example, the word MQM has an edit distance of 1 with valid words MOM, MAM, MUM. Since we can learn that P3 often confuses Q with O (the nature of the strokes are similar), but hardly confuses Q with A or U, a P3-aware spelling correction tool can suggest MOM with high confidence. A less trivial

example occurs when P3 outputs, say, NIET. Words NET and MET have edit distances of 1 and 2, respectively. However, the spelling tool could observe that P3 confuses "M" as "NI" with far greater probability than "E" as "IE". Thus, one could predict that the user intended to write MET with reasonably high probability, even though its edit distance is higher. Formally, given a mis-spelled word w, the P3-aware spelling corrector computes the word ϕ_w as follows.

$$\phi_w = \{i : \frac{P(w|i)}{P(w|j)} > 1\} \ \forall \ valid \ words, i, j, i \neq j$$

The distribution of $P(w|i)$ is learned from our own data set, and can adapt to the user's idiosyncrasies over time. We have implemented this tool and found improvement over dictionary based spelling correction.

4.5 Control Gestures

To write a short phrase, the words need to be separated by spaces. In certain cases, the characters may need to be deleted. Further, the user should be able to email the written/drawn content to her email address. These are a few control operations that are vital to improve the user's experience. The PhonePoint Pen assigns a unique gesture to each of these, and recognizes them without difficulty. Specifically, the space is denoted by a long horizontal movement or two dots. The deletion is like using an eraser – the users shakes her hand at least four times briskly. To email, the user must draw a check mark in the air. With these functionalities in place, we present the implementation details of P3, followed by performance evaluation.

5. IMPLEMENTATION AND EVALUATION

We prototyped the PhonePoint Pen on a Nokia N95 phone. The 3D accelerometer obtained 30-35 acceleration readings per second. We developed a server side implementation in MATLAB. Basic MATLAB libraries allowed us to implement signal processing techniques (low pass filtering) and simple statistical analysis. We prototyped this code on Python for on-phone processing, thus users write in air and the output is shown on the screen. The current Python implementation supports writing only one character at a time. To port P3 to Python, some of the techniques were simplified (filtering operations modified to running averages and subtractions). The results from Python and MATLAB differed in few instances. The following sections report results obtained when processing the accelerometer readings in MATLAB.

The remainder of this section is organized in three parts: (1) evaluation metrics and methodology, (2) PhonePoint Pen evaluation with students, and (3) experiences from patients with cognitive/motor impairments conducted at the Duke University Hospital.

5.1 Evaluation Metrics

The P3 evaluation is centered around character and word recognition accuracies. We define *Character Recognition Accuracy (CRA)* as the fraction of successful typed text recognitions, when a user writes individual alphabets/characters (used interchangeably). In addition to CRA, we also evaluate P3's quality of geometric representation. For this, we display the geometric characters to a human, and ask her to recognize them. The correctly identified fraction is defined as the *Human Readability Accuracy (HRA)*.

To compute *Word Recognition Accuracy (WRA)*, we randomly generated English words from a dictionary and requested test users to write them in air. Longer words are naturally more prone to mistakes because every character and every transition will have to be precisely decoded. Thus WRA degrades with word-length. Nevertheless, since P3 outputs typed-text we can apply spelling correction to improve the final accuracy. Thus, we report WRA for basic P3, WRA with *English-Spelling-Correction*, and WRA with *P3-Aware-Correction*. We also report WRA with Human Readability (i.e., fraction of words correctly recognized by humans).

5.2 Evaluation Methodology

We conducted PhonePoint Pen tests mainly with students from computer science and engineering. The test group comprised of 10 students in two categories: Trained and Novice. Novice students (6/10) were defined as users that practiced less than 10 characters before starting the evaluation. The rest were Trained students who practiced 26 characters (each English alphabet approximately once, taking less than 5 minutes in total). Only one of the Trained users rehearsed for 75 characters before starting the tests. Besides university students, we also performed a study with a small population of 5 patients from the Duke University Hospital – the primary purpose was to gain insights into P3's applicability into assistive technology. According to our IRB approval, the patients were allowed to write up to 8 characters. The patients had no previous experience with our prototype, and performed the experiments under the supervision of care-givers. Although P3 broadly failed in the tests (due to occurrences that the system was not designed for), we will report the valuable experience and feedback we gained from neurosurgeons, physicians, and speech pathologists. We will discuss the modifications we have made to P3 based on these real-life feedback.

5.3 Performance Evaluation

The main evaluation results are summarized as follows.

- Figs. 7 and 8 show sample words written with P3. Fig. 9 quantifies this by showing that average readability is 83% and 85.4% for characters written by Trained and Novice writers respectively. Employing stroke grammar, average character recognition is 91.9% and 78.2% for the same two groups (Fig. 10). Numeric digits experience similar accuracy.

- Zooming into results, most Novice users achieve similar CRA to Trained users (Fig. 11) – 2 weakest users achieve comparable accuracy by writing on a table-top (Table 1). Disambiguation approaches are reasonably effective (Fig. 12). However, the average character writing time is between 3.02 to 4.3 seconds (Fig. 13), the main limitation with the current system. Energy consumption with P3 is not a concern (Fig. 14).

- Word recognition degrades with increasing word-length, however, spelling correction helps – 80% for 5 letter

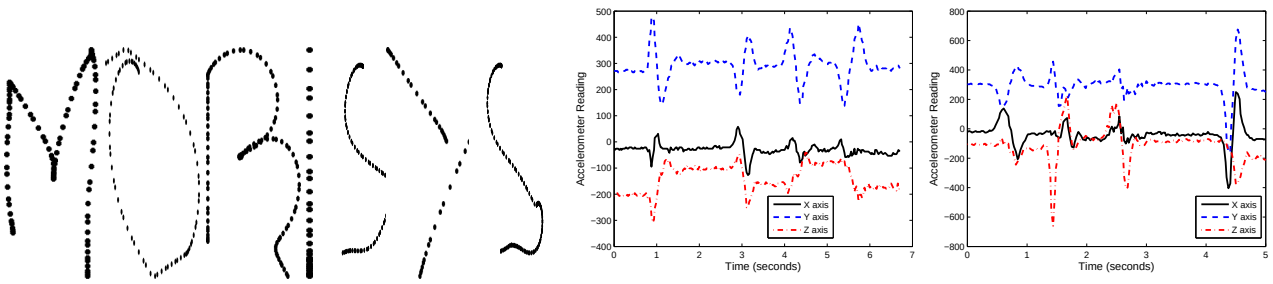


Figure 7: Alphabets M, O, B, I, S, Y, S, as outputs of the PhonePoint Pen. Although distorted, the characters are legible. The raw acceleration data is shown for the Alphabets M and Y.

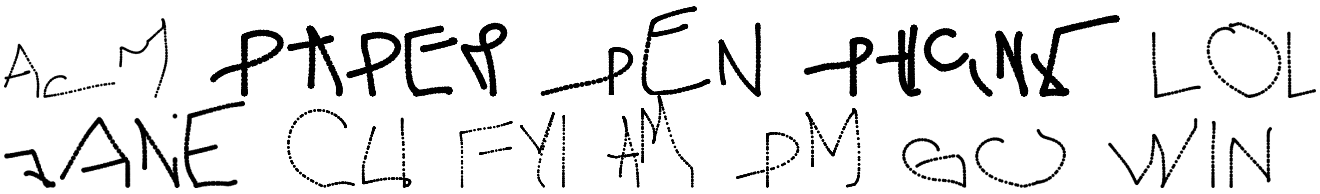


Figure 8: Samples of air-writing (few SMS lingo) recognized correctly by stroke grammar: ACM, PAPER, PEN, PHONE (not legible), LOL (laugh out loud), JANE (a name), CU (see you), FYI, AM, PM, GO and WIN.

words (Table 2). Human readability is lower: 70% for 5 letter words. Tests with hospital patients show low accuracy due to usability issues (Table 3). Advised by speech surgeons, we emulated a class of patients by writing with the left hand – P3 achieves 81.7% accuracy (Fig. 15).

Results: Readability and Recognition

Figure 7 shows the geometric version of alphabets M, O, B, I, S, Y, and S, written by a trained user (each alphabet written separately). The acceleration readings for E and Y are presented alongside. P3 correctly converts the acceleration to alphabets, while the geometric versions are human legible. Figure 8 shows some examples of air-written words. Evidently, the lack of a reference frame degrades the sense of proportion and relative placement of characters. Despite these distortions, the stroke grammar yielded correct results for all the words in Figure 8.

Towards a systematic evaluation, Figure 9 shows the *Human Readability Accuracy (HRA)* and the *Character Recognition Accuracy (CRA)* per-alphabet, per-user-category (526 characters were written in total). For HRA, each of the 526 alphabets were presented to arbitrarily selected students. Average HRA for Trained and Novice categories proved to be 83% and 85.4% respectively. This is likely because human cognition is powerful and is able to decipher even highly distorted characters (the key intuition with Captchas [26]). Thus, even though Novice users exhibited greater distortion in the geometric alphabets, human readability for both the categories remained comparable. The expected difference between the two categories became evident in the CRA comparison. Figure 10 shows an average of 91.9% and 78.2% CRA for Trained and Novice categories, respectively. This suggests that 2-4 minutes of training has a positive impact – users learn how the system reacts to their hand-movements and adapt somewhat involuntarily. Numeric digits achieved comparable accuracies (not shown in the interest of space).

Results: Per-User Accuracy

The accuracy of Novice category users was relatively lower – the following discussion zooms into the results. We re-plotted the results from the experiments on a per-user basis (Figure 11), and observed that the variance among the Novice users was quite high. Four novice users were able to achieve reasonably good CRA (in fact one of them was better than the Trained users), while two other novices were not able to exceed a CRA of 70%. In response to this finding, we measured how these users performed when writing on a flat surface, like a table-top. Our hypothesis was that certain involuntary 3D hand motion, or intra-stroke wrist rotation, is likely to affect recognition – writing on a physical surface could improve performance.

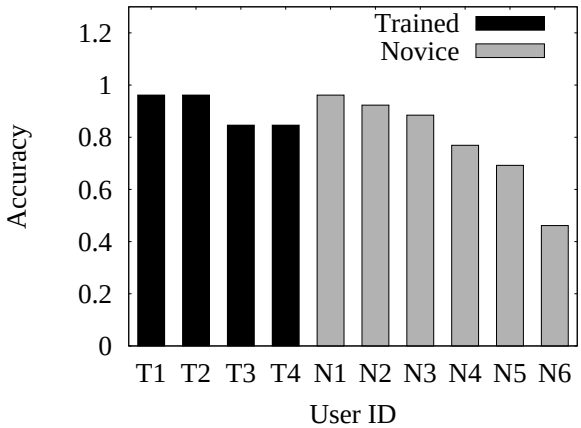


Figure 11: Average CRA per-user. Novice users have a wide variation in accuracy.

Table 1 shows the accuracy improvement when the two weak Novice users wrote the alphabets on a table-top. For instance, while writing "A", the phone touched the table for the "/\"

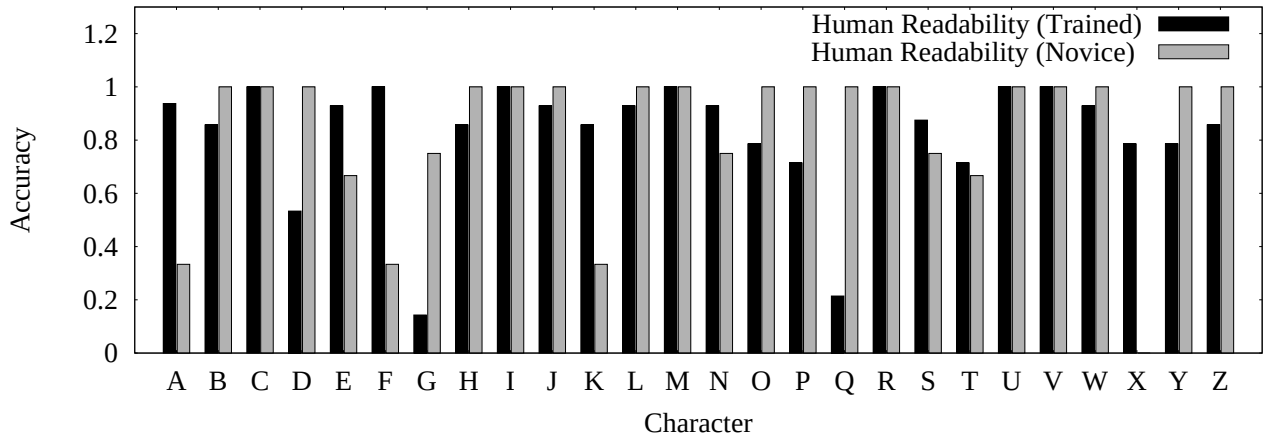


Figure 9: Human Readability Accuracy (HRA) for Trained and Novice users. Human's powerful cognitive abilities result in comparable recognition performance, even though the Novice's characters were qualitatively observed to be more distorted.

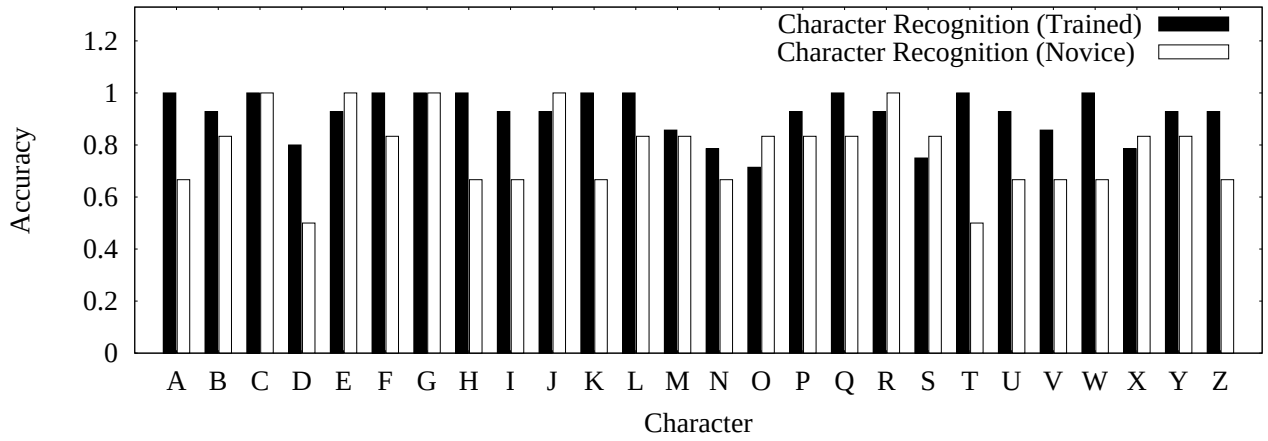


Figure 10: Character Recognition Accuracy (CRA) for Trained and Novice users per-character. The stroke grammar achieves a reasonably high accuracy, especially with Trained users.

strokes, was lifted and repositioned on the table again, and then the user wrote the "–". The on-table accuracy improved substantially for the users; the weakest user experienced a jump from 46.2% to 76.9% CRA. Assuming that flat surfaces are often accessible, P3 may be acceptable even to the weak user.

Table 1: Writing In-Air Vs. On-Table

User	In-Air	On-Table
Weakest User	46%	77%
2nd Weakest User	69%	88%

Results: Grammar Disambiguation

Recall that the P3 stroke-grammar exhibits inherent ambiguity. For example, D and P are written with the same strokes "|" and "⊃". Similarly, characters {V, X, Y} and {O, 6, S} use common sets of strokes. We disambiguate between these characters by looking at the directions of the stroke (V has a downward followed by a upward movement, while X has a downward, pen reposition, and then again downward). We also track the movement of the phone during the pen-repositioning phase to extract more information about the user's intention. Figure 12 presents the accuracy of disambiguation, along with the actual outcomes when the disambiguation fails. In majority of the cases, the character is decoded correctly. Also, among incorrectly decoded characters, some are not confused with their ambiguous counterparts (e.g., H and U). However, ambiguity still occurs, e.g., 0 with 6 in 28.6% and 6 with 0 in 35.7% cases; P with D in 7% cases.

Results: Writing Speed

Figure 13 presents two CDF curves. One denotes the distribution of alphabet-writing time computed across all users. The median time was 4.3 seconds. We believe users displayed a tendency to write slower than necessary, partly because they were new to the system, and because they were keen on optimizing for accuracy. P3, however, can support quicker writing. To understand the speed limits with P3, we measured the minimum time incurred in writing each alphabet correctly. The second curve plots this distribution, and evidently, the median improves to 3.02 seconds. Nevertheless, even the best performance of P3 is quite slow, and is currently the key limitation with the prototype. We believe a number of opportunities exist that will increase the speed of air-writing.

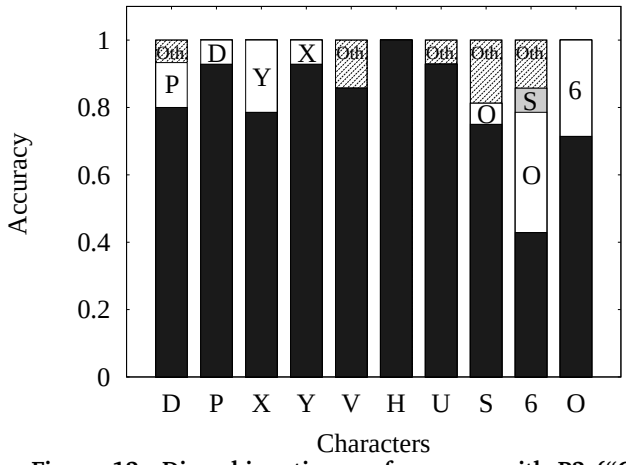


Figure 12: Disambiguation performance with P3 ("Oth" in the striped portions implies "Others").

Greater number of accelerometer samples per second is an immediate one; we will discuss others as a part of our ongoing work.

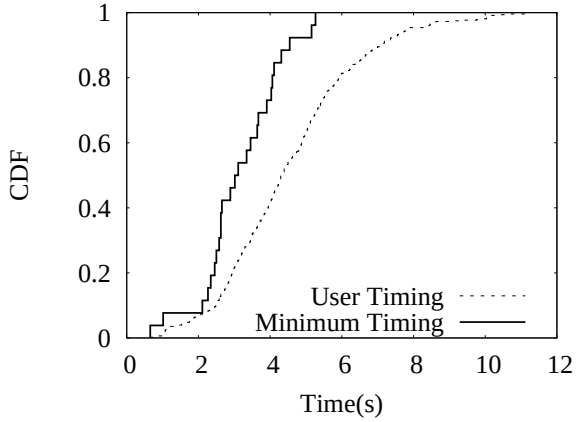


Figure 13: Distribution of time to correctly write English characters with P3.

Results: Word Recognition

We requested users to also write English words in air (we did not include the two weakest users for these experiments). The words ranged from 2 to 5 characters and were chosen randomly from a dictionary. The users wrote 20 words for each word-length. Table 2 reports P3's Word Recognition Accuracy (WRA). Spell checking with an English dictionary improves the accuracy; P3-aware corrections led to further improvements. This is because the P3-Aware spell checker better understands the relationship between user-intent and P3-output, and is able to make the necessary corrections. We also observed reasonable performance through human recognition.

Results: Energy Measurements

We ran experiments to compute the energy footprint of the Nokia N95 accelerometer. We sampled the accelerometer at the same rate as PhonePoint Pen on a fully charged Nokia N95 8GB phone (Figure 14). The phone exhibited an average battery-lifetime of 40 hours, i.e., a user should be able to continuously write for 40 hours with P3.

Table 2: Word recognition.

Word Length	Phone Pen	Spell Correct	P3-Aware Spell Correct	Human Readable
2	17/20	19/20	20/20	11/20
3	18/20	19/20	19/20	12/20
4	13/20	18/20	19/20	10/20
5	13/20	16/20	17/20	14/20

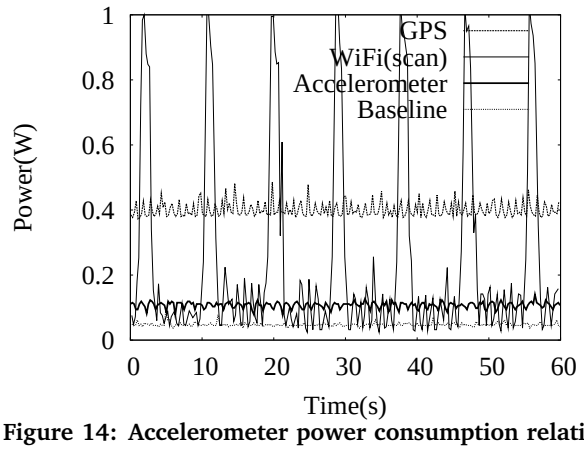


Figure 14: Accelerometer power consumption relative to other phone sensors.

5.4 Experiences with Hospital patients

In collaboration with Physicians from the Surgery/Speech Pathology department of Duke University Hospital, we carried out PhonePoint Pen tests with patients suffering from various forms of cognitive disorders and motor impairments. Based on IRB approval, 5 patients were approved of writing 8 randomly chosen alphabets. The patients were selected with varying degree of motor-impairments (e.g., a hydrocephaleus lumber drain trial patient exhibited good cognition but weak motor skills; a patient from a major car accident, 12 days prior, had a right side paralysis and spinal injury, but was able to write with his right hand; a 72-year old stroke patient had weakness on both limbs with severe tremors). The tests were carried under the supervision of medical practitioners and care-givers, who first learned to use P3 from us. We were not allowed to observe the patients, however, we interacted closely with the care-givers to receive feedback. Importantly, P3 generated wide interest in the hospital, drawing neurosurgeons, speech therapists, and care-givers to witness the tests and comment on the potential applications and additional requirements. The overall experience proved to be invaluable. We report the main lessons here.

(1) The P3 design requires users to press a button to start the application, and to re-press the button to stop. While this did not appear as an important design issue among university students, it proved to be a bad design choice for assistive technology applications. Patients found air-writing quite intuitive, but were unable to press the button appropriately. One patient pressed the buttons many times, another pressed the wrong one, and yet another found it hard to press. Table 3

shows the results. The unanimous advise from doctors was to eliminate button presses. Based on this feedback, we implemented the start and stop of the system through shaking. A user shakes the phone in air before beginning to write, and then shakes again to stop.

Table 3: Patient performance.

Patient ID	1	2	3	4	5
Accuracy	1/8	1/8	1/8	5/8	could not press button

(2) A neurosurgeon criticized that the P3 prototype required "shoulder, elbow, and wrist coordination", a constraint that may be difficult to satisfy by hospitalized patients. His recommendation was to reduce the size of the letter so that it can be written with elbow movements alone. Moreover he suggested developing filters that would cancel the tremor in people's hands, and thereby recognize the characters. Our ongoing work is focused on learning the natural tremor of the person, and suitably canceling it from the entire accelerometer signal.

(3) One particular advantage of P3, even in light of specialized medical gadgets, is familiarity with cell phones. Physicians and care-givers emphasized the difficulties patients face in adopting new technological gadgets, particularly at the higher age group. Using the patient's own phone to gesture "made a lot of sense". They said, with a good degree of reliability, they envision a wide range of applications. Interestingly, several nurses showed enthusiasm at the prospect of the patient changing her TV channel by writing in the air (pressing the remote control button is again a difficult task for many).

(4) Exhaustive tests with patients are difficult due to IRB restrictions; the turn around time is also high. To partly overcome this, doctors suggested that it is valuable to test P3 by having normal right-handed users write with their left hand. Several speech-impaired patients only have minor problems with hand motions, and left-handed writing may be a credible emulation of these conditions. Figure 15 shows the CRA for consonants and vowels when 4 users wrote all the alphabets left-handed (we do not include per-alphabet accuracy in the interest of space). The average accuracy across all the alphabets and all users is 81.73%.

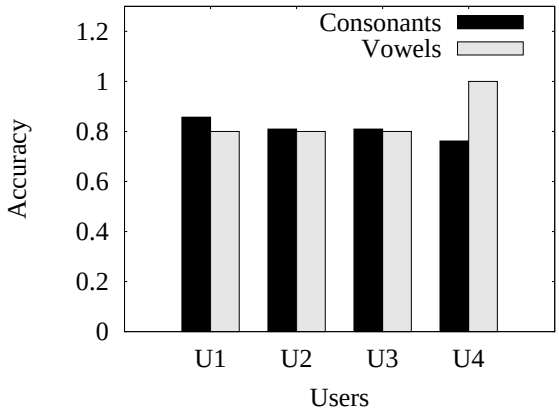


Figure 15: Accuracy with left-handed operation emulating speech-impaired patients.

6. LIMITATIONS

Developing the PhonePoint Pen to the standards of a commercial product calls for further research. Nevertheless, we believe our current prototype has made credible advances towards the end goal. This section discusses the current limitations and opportunities for further improvement.

Quicker Writing
The main limitation with P3 is its speed of writing – at best 3.02 seconds per alphabet on average[2]. Writing faster degrades accuracy. This is because the accelerometer exports around 30 samples per second, and is therefore inadequate to capture all the motions (especially in multi-stroke alphabets like E, H, W, etc.). While higher sampling rates will certainly be valuable [31, 32], we are investigating the benefits of using two accelerometers (available in OpenMoco phones). Further, we believe that there are opportunities in using the gyroscope and camera to determine hand-movements. By observing the changing camera-view over time (when the user gestures in air), the alphabets may be captured quicker. Gyroscopes are expected to alleviate the problems with wrist-rotation.

Writing Long Words and Drawing Pictures
Drawing capabilities require sophistication. The main problem stems from the difficulty in tracking the phone movement while the pen is being repositioned in 3D space. Thus, although the actual written words/shapes are identified, their relative placements are often incorrect. The problem is pronounced when the figure involves multiple pen-repositioning. Long words and sentences face similar problems. We leave these solutions to future work.

Cursive Handwriting
Supporting cursive writing is certainly desirable with P3, however, significantly more difficult to accomplish. The problems of stroke-detection and character-recognition are exacerbated due to the continuous movement of the hand. One approach would be to apply pattern recognition algorithms on the entire stream of accelerometer readings. However, such a scheme will not only require complex computation, but may also need a non-marginal degree of training. We have traded off this functionality for simplicity.

Writing while Moving
If a person writes in the air while moving, the accelerometer readings will reflect that movement. Our current prototype assumes that the user is stationary while writing. Characterizing user motion, and subtracting them from the accelerometer signals, is a topic of future work.

Comment on Survey and Testing Population
Students who have tested P3 are mostly students from the Computer Science and Engineering. These students are likely to have an understanding of accelerometers, and could have adapted to P3's behavior. In that sense, our accuracy results for Novice users could be partially optimistic. Nevertheless, with a little bit of training, even lay users should be able to adapt to P3.

[2]Of course, this should perhaps not be compared against typing on a keyboard or writing running hand.

Greater Algorithmic Sophistication
Bayesian Networks [4] and Hidden Markov Models (HMMs) [31, 32] have been successfully used for gesture recognition. These approaches are powerful and certainly applicable to PhonePoint Pen (as opposed to a simple grammar). The down side is that these models require meticulous feature selection and extensive training to produce good estimations. Training may be inconvenient if the user is required to do this operation herself [18]. The alternative approach is to carefully prebuild the models to ensure broad applicability among users. Hardware heterogeneity needs to be accounted for as well, since the phone OS or the accelerometers themselves may provide various sampling rates across different phone models. Finally, we observe that the P3 tree-based grammar can be retrofitted to HMMs. The sequences of strokes can be regarded as HMM states representative of a character. Transition probabilities between states (characters) may be computed through training. Subsequent strokes written by the user will advance the HMM states and help infer the user scribble (e.g., adding a "|" to an N may be indicative of a highly probable M). We plan to investigate these techniques as part of our future work.

7. RELATED WORK

Designing an alternative input technology is a rich area of research. Numerous sensors on mobile devices fuel this area to rapid growth. Naturally, a large body of existing work relates to the PhonePoint Pen. We touch upon these works briefly (in the interest of space), while discriminating the contributions from this paper.

Air-gestures with 3D accelerometers.
Gesture recognition has been widely studied through accelerometers, gyroscopes, vision based techniques, etc. [31, 32, 8, 9, 3]. Works that are closest to P3 include (1) a sensor mote-based 4-character recognizer [32], (2) a numeric digit recognizer with customized hardware [25, 31], and (3) *uWave*, a mobile phone based single-gesture recognizer. The first two works employ a highly capable accelerometer (around 100 samples/s). They use Principle Component Analysis (PCA), Hidden Markov Models (HMMs), and Dynamic Time Warping (DTW) algorithms, to achieve accuracies of 90 to 94%. However, the accuracy falls to 80% when the accelerometer is sampled at 40 samples/s. More importantly, the proposed systems are only able to write few numeric digits, *that do not require the user to reposition the pen within the same character.* Geometric figures are also not viable because gestures are identified through pattern matching, and hence, *the system does not compute the actual displacement and direction of motion.* PhonePoint Pen, on the other hand, tracks the user's hand movement, and develops methods for pen-repositioning, character transition, stroke-grammar, rotation avoidance, and character disambiguation.

uWave [14] is a mature work that allows a user to gesture with mobile phones, enabling simple operations like gesture-based user authentication, opening/closing applications, etc. The authors attain an impressive 99% accuracy with 8 gestures and negligible training. While this is valuable for a number of interfacing applications, we emphasize that character recognition entails an additional set of problems. Specifically, gestures are significantly tolerant to error; as long as the errors repeat across all gestures, the gesture can be identified. In contrast, the PhonePoint Pen requires a different approach to continuously track a more complicated motion of the hand.

Vision based gesture recognition.
Cameras have been used to track an object's 3D movements in the air [20]. TinyMotion [27] uses image sequences captured by the built-in camera to detect the movements of the cell phone. Movements are limited to horizontal, vertical, rotational, and phone tilt. Converting these movements to characters and words introduces additional challenges. Nevertheless, these movements may be fused with the accelerometer output and result in more reliable stroke identification.

Microsoft Research recently demonstrated a project titled "write in air" [2], that uses an apple in front of a camera to air-write alphabets. Computer vision based algorithms can precisely discern the movement of the apple (or any other object) to create both geometric and textual representations of the alphabets. Noisy accelerometers and limited processing in mobile phones lack several advantages present in computer-connected cameras. Moreover, the system does not recognize words, side-stepping the problems of transition between characters. Signal processing based techniques are useful here, but not sufficient [1, 13, 29]

Stylus-based sketch recognition.
A number of systems inferred user sketches drawn with a stylus on a pad or Tablet PC. SketchREAD [4] aimed to identify the parts of electrical diagrams and family trees drawn by users. The system understands the user sketch based on a list of basic symbols, a hierarchical representation of symbol combinations, and a dynamically constructed Bayesian network. By employing a Bayesian network, SketchREAD can accommodate errors in low-level symbols (e.g. if a line is identified as an arc). The Bayesian network is trained and learns such likely mistakes. While similar in spirit with our grammar-based tree, the Bayesian network is a powerful paradigm and can improve P3 performance. We plan to investigate Bayesian approaches as part of P3 future work.

Another project titled the Electronic Cocktail Napkin [11] developed a sketching environment in which user diagrams can be identified and interpreted by the program. The system takes advantage of the pad on which the user is sketching. It identifies pen paths, number of strokes, corners, sketch size, aspect ratio, and rotation. These features are then matched against a library of templates. The templates are learned by allowing the user to draw several examples and save them into the system. Unlike the Electronic Cocktail Napkin, P3 operates on an imaginary blackboard which makes extracting similar features difficult. Without a reference frame, sketch size, aspect ratio, and rotation are hard to approximate. Further, P3 uses a phone accelerometer to infer strokes, which unlike a stylus, is more noisy. Nevertheless, the Electronic Cocktail Napkin identifies a space of possible features that we plan to explore in P3.

Xerox PARC also investigated stylus-based interactions in a project called Unistrokes [10]. In their approach, the user is required to learn a Unistroke alphabet which maps each English character to a symbol written with one stroke (one continuous movement). The alphabets that require lifting (e.g., A, F, E, K) are simplified to be drawn with one stroke.

Words are separated by a special character (a period "."). Once users get accustomed to Unistroke writing, they can achieve speeds of 1 to 1.8 characters per second. Similar to Unistroke, Graffiti [17] uses single stroke character alphabets to allow handwriting recognition. Graffiti characters are designed to look close to their associated English alphabets, and thus be more user-friendly than Unistrokes.

Many pen-touch based Tablet PCs offer built-in handwriting recognition software. In general, these solutions use both geometrical and temporal information to infer the written characters. The user input is considered as a sequence of dots, function of time. The dots order is used to identify basic strokes such as circles and lines, which are further used to identify either characters or complete words. Velocity information may serve to infer transitions between characters/strokes (the user writing naturally slows down when switching directions – similar to the pausing time between characters required in P3). On top of this low level information, handwriting recognition employs Hidden Markov Models. Further, as in P3, a vocabulary is used to rectify words (e.g., closest edit distance). Optionally, when writing sentences, semantic information is employed to improve over vocabulary-based word accuracy. We note, that unlike P3, writing recognition on a Tablet PC does not use acceleration to compute stroke displacement. In the Table PC case, the displacement values are accurate, resulting from the mechanical touch of the pen on the Tablet's screen. Thus, the stroke information is more reliable than the one used in P3 (through integration of the phone accelerometer reading). Further, the Tablet screen provides a reference frame, and thus the relative positions of strokes can provide additional clues. P3 does not benefit from a reference frame. Instead, P3 needs to decide the stroke placements and form the user intended character.

Optical Character Recognition (OCR) is a technique for transforming text represented in image format (e.g., bitmaps) into typed text (e.g, ASCII characters). OCR relies on matrix matching to infer characters. First, OCR scans the text image which results in a set of dot matrices. Each dot matrix is then compared against template matrices (for each character) part of the OCR library, and the best match yields the typed character. Other approaches for OCR use features of the scanned input to infer the typed text. Features include shapes, lines (horizontal, vertical or diagonal), line intersections and spacings between shapes. Matrix matching is preferred when the character type (e.g., font) in the image does not vary. On the other hand, feature-based OCR accommodates multiple styles of text. Note, that in general OCR input is "well" formatted text, mostly resulting from image-scanned text (e.g, book scans). Unlike OCR, P3 operates on much noisier input.

Wiimote, Logitech Air-Mouse, and Nokia NiiMe.
A popular device capable of tracking hand movement is the Wii remote (or "Wiimote") used by the Nintendo Wii console [20]. The Wiimote uses a 3-axes accelerometer to infer forward and backward rapid movements. In addition, optical sensors aid in positioning, accurate pointing, and rotation of the device relative to the ground. The optical sensor is embedded on the Wiimote and relies on a fixed reference (a sensor bar) centered on top of the gameplay screen. The "Wiimote" can be augmented with the "Wii Motion Plus", a pluggable device containing an integrated gyroscope to cope with hand rotation. These three sensors – the accelerometer, the gyroscope, and the optical sensor – can reproduce motions similar to real arm-motion. Similarly, the PlayStation Move [23] is equipped with accelerometer and gyroscope sensors. Further, a digital camera and a LED orb at the top of the controller, aid in tracking the user gestures. Unlike these devices, the Nokia N95 consists of only a (low-cost) accelerometer, and limited processing capabilities. Developing P3 on a mobile phone presents unique research challenges. Nevertheless, the arrival of gyroscopes in consumer phones will address and solve some of the current P3 challenges (hand rotation) and facilitate less constrained air-writing (removal of pauses between strokes). We plan to integrate the gyroscope in the next implementation of PhonePoint Pen.

The Logitech Air Mouse [16] targets people who use computers as multimedia devices. The Air Mouse provides mouse-like functionalities but the device can be held in air similar to a remote control. Accelerometers and gyroscopes together allow for linear and rotational motion of the pointer on the screen. Unlike the Air Mouse, P3 does not have a screen on which the human user may see and adjust the pen movement in real time. Thus P3 must estimate relative position of strokes and characters without any visual reference frame.

The NiiMe [5] project transformed the Nokia N95 phone into a bluetooth PC mouse. The PyAcceleREMOTER project developed a remote control for the Linux media player MPlayer. By tilting the phone, the player's play, stop, rewind, fast-forward, etc. are controlled. Other applications like *Inclinometer* provides car inclination while *Level Tool* allows measurement of the inclination of different surfaces by placing the phone on that surface. Lastly many video games for the N95 phone make use of the accelerometer, e.g., to guide a ball through a maze. Being able to write in the air, we believe, is a more challenging problem.

Smart Pen and SmartQuill.
Livescribe Smartpen [15] is a pen-like device capable of tracking a person's writing. The device requires a special finely dotted paper to monitor the movement of the pen. The pen recognizes alphabets which can be downloaded to a PC. However, the dotted paper may not be always accessible, making ubiquitous note-taking difficult. Tablet PCs also suffer from this problem of ubiquitous accessibility. SmartQuill [28] is a pen device intended to recognize handwriting without the need of a special pad. It can use any surface for writing including on air. SmartQuill requires significant per-user training. If the training and test user differ, word recognition accuracy is severely affected [19]. Unlike these approaches, PhonePoint Pen does not rely on special hardware or paper, and does not require training.

8. CONCLUSIONS

This paper attempts to exploit the accelerometer in mobile phones to develop a new input technology. While today's users are mostly used to keyboards and touch-screens, we propose to mimic a pen. By holding the phone like a pen, the user should be able to write short messages in the air. The phone identifies the hand gestures as one of multiple strokes, compares the sequence of strokes against a grammar, and recognizes the air-written alphabets. The entire process

requires negligible practice, and owing to its algorithmic simplicity, can run entirely on the phone's processor. The written message is displayed on the phone-screen, and may also be emailed to the user if desired. We believe that in the age of microblogging and tweeting, such input devices may be effective to note down information on the fly. Moreover, the pen may offer an intuitive user-experience, adding to the menu of current input methods. We call this system PhonePoint Pen, and demonstrate its feasibility through a Nokia N95 prototype and real user studies. The performance results are promising, while the user feedback (from the student community) is highly positive.

9. ACKNOWLEDGMENTS

We sincerely thank our shepherd, Mark Corner, as well as the anonymous reviewers, for their invaluable feedback. We express sincere gratitude to the various doctors, nurses, and care-givers at Duke Hospital who took time from their busy schedule to help in the P3 tests. We also thank the volunteers, including Xuan Bao, Justin Manweiler, Naveen Santhapuri, Souvik Sen, and several others, who volunteered to write in air for P3 evaluation. Finally, we are grateful to NSF for partially funding this research through the NSF IIS-910846 grant, as well as to Nokia for providing with Nokia N95 phones.

10. REFERENCES

[1] AiLive LiveMove pro. AiLive Inc. http://www.ailive.net/liveMovePro.html.

[2] Microsoft. Write in The Air, TechFest 2009. http://www.youtube.com/watch?v=WmiGtt0v9CE.

[3] S. Agrawal, I. Constandache, S. Gaonkar, and R. Roy Choudhury. Phonepoint pen: using mobile phones to write in air. In *ACM MobiHeld*, 2009.

[4] C. Alvarado and R. Davis. Sketchread: a multi-domain sketch recognition engine. In *ACM UIST*, 2004.

[5] A. Arranz. Niime. http://www.niime.com/.

[6] V. Balakrishnan and P. H. Yeow. Sms usage satisfaction: Influences of hand anthropometry and gender. In *Human IT 9.2*, 2007.

[7] V. Balakrishnan and P. H. Yeow. A study of the effect of thumb sizes on mobile phone texting satisfaction. In *Journal of Usability Studies*, 2008.

[8] T. Baudel and M. Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 1993.

[9] X. Cao and R. Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. *ACM Trans. Graph.*, 2004.

[10] D. Goldberg and C. Richardson. Touch-typing with a stylus. In *ACM CHI*, 1993.

[11] M. D. Gross. The electronic cocktail napkin–a computational environment for working with design diagrams. *Design Studies*, 1996.

[12] K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *ACM CHI*, 2005.

[13] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Comput.*, 2006.

[14] J. Liu, Z. Wang, and L. Zhong. uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications. Mar. 2009.

[15] LiveScribe. Smartpen. http://www.livescribe.com/.

[16] Logitech. Air mouse. http://www.logitech.com.

[17] I. S. MacKenzie and S. X. Zhang. The immediate usability of graffiti. In *Graphics Interface*, 1997.

[18] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, 2004.

[19] B. Milner. Handwriting recognition using acceleration-based motion detection. In *Document Image Processing and Multimedia, IEEE Colloquium on*, 1999.

[20] Nintendo. Wii console. http://www.nintendo.com/wii.

[21] Nintendo. Wii motion plus. http://www.nintendo.com/whatsnew.

[22] Nokia. Virtual keyboard. http://www.unwiredview.com/wp-content/uploads/2008/01/nokia-virtual-keyboard-patent.pdf.

[23] PlayStation. Move. http://us.playstation.com/ps3/playstation-move/.

[24] C. Soriano, G. K. Raikundalia, and J. Szajman. A usability study of short message service on middle-aged users. In *OZCHI*, 2005.

[25] C. Sung-Do, L. A.S., and L. Soo-Young. On-line handwritten character recognition with 3d accelerometer. *International Conference on Information Acquisition*, 2006.

[26] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Commun. ACM*, 2004.

[27] J. Wang and J. Canny. Tinymotion: camera phone based interaction methods. In *CHI '06 extended abstracts on Human factors in computing systems*, 2006.

[28] L. Williams. Smartquill. http://sites.google.com/site/sensecam/smartquill.

[29] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *ACM UIST*, 2007.

[30] Xbox. Kinect. http://www.xbox.com/en-US/kinect.

[31] S. Zhang, C. Yuan, and Y. Zhang. Handwritten character recognition using orientation quantization based on 3d accelerometer. In *Mobiquitous*, 2008.

[32] D. Zhuxin, U. C. Wejinya, Z. Shengli, S. Qing, and W. J. Li. Real-time written-character recognition using mems motion sensors: Calibration and experimental results. In *ROBIO*, 2009.