An Energy Harvesting Wearable Ring Platform for Gesture Input on Surfaces

Jeremy Gummeson[1,2], Bodhi Priyantha[2], Jie Liu[2]
[1]University of Massachusetts Amherst, [2]Microsoft Research
gummeson@cs.umass.edu, bodhip@microsoft.com, jie.liu@microsoft.com

ABSTRACT

This paper presents a remote gesture input solution for interacting indirectly with user interfaces on mobile and wearable devices. The proposed solution uses a wearable ring platform worn on a user's index finger. The ring detects and interprets various gestures performed on any available surface, and wirelessly transmits the gestures to the remote device. The ring opportunistically harvests energy from an NFC-enabled phone for perpetual operation without explicit charging. We use a finger-tendon pressure-based solution to detect touch, and a light-weight audio based solution for detecting finger motion on a surface. The two-level energy efficient classification algorithms identify 23 unique gestures that include tapping, swipes, scrolling, and strokes for hand written text entry. The classification algorithms have an average accuracy of 73% with no explicit user training. Our implementation supports 10 hours of interactions on a surface at 2 Hz gesture frequency. The prototype was constructed with off-the-shelf components and has a form factor comparable to a large ring.

Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: [Real-time and Embedded Systems]

Keywords

Near Field Communications; Energy Harvesting; Wearable Computing; Gesture Input; Machine Learning

1. INTRODUCTION

Increasingly, users interact with their mobile devices on the go. Many of us listen to music on mobile devices while traveling, we constantly check our E-mails, shoppers browse through their shopping lists and do price comparisons while shopping, and people constantly access applications such as Facebook, Twitter, and Foursquare.

MobiSys'14, June 16–19, 2014, Bretton Woods, New Hampshire, USA.

http://dx.doi.org/10.1145/2594368.2594389 .

Interacting with mobile devices on-the-go requires the user to enter different gestures to scroll, zoom, flip, and enter text on UI elements. The mobile phone with its relatively large display has provided a unified and convenient platform for such interactions. However, more recent trends in wearable devices such as glasses, wrist bands, and watches have made such interactions limited and awkward due to the lack of touch real estate and the positioning of the device itself [5].

While mobile device interfaces continue to shrink, interfaces of remote display devices such as TVs and game consoles are becoming even more complex, requiring extensive maneuvering via simple remote controllers or requiring remote control with full keyboard-like capability [6]. For example, with a conventional remote control, a simple task such as entering text to search for a movie title becomes a monumental task leading to a poor user experience. The increasing user mobility and availability of ubiquitous computing and entertainment resources will create a strong demand for light-weight solutions that enable rich interaction with remote displays and augmented spaces.

In this paper we present a ring form-factor wearable device for gesture input. The user interacts with an arbitrary surface; such as a desk, chair arm rest, or the users clothing; with the ring-worn finger to enter multiple gestures. Our platform supports a rich set of gestures similar that found on a modern touch based UI. These gestures include "tap" for selection, "flip" gestures that quickly flip through multiple windows or items in a list, "scroll" for scrolling through a web page or a list of items, and writing-based "text-entry". Altogether, our platform can interpret 23 unique gesture primitives made on an arbitrary surface.

Some previous solutions for motion based gesturing support only a limited number of gestures such as a few patterns [13], or only text-entry under constrained conditions such as large character sizes [1] and are implemented on resource rich devices such as cell phones. Other solutions for gesturing on surfaces either required instrumented surfaces [11]; while others use heavy processing such as MFCC computations on audio, and power hungry sensors such as gyroscopes [21] which are impractical on resource limited devices such as a self-contained ring platform. Solutions that instrument the finger tip [20] are inconvenient due to the impact on the user's touch.

Our goal is to develop a wearable platform, which is likely to be adopted by a wide user community, that provides always-available gesture input using gestures typically used in modern touch-based input devices. Developing such a wearable device poses several challenges:

The first challenge is achieving the right balance of usability and performance. A large heavy platform with ample resources would be uncomfortable to wear on a finger continuously, limiting the usability of the device; while a platform with a form factor and weight similar to a typical ring is an ideal platform for achieving the goals of wide adoption and availability.

However, these form factor and weight limitations severely limit the energy and computational resources available on the ring; to make matters worse, the ring needs to use an expensive wireless link to transfer information to a remote device. We use carefully designed sensing techniques, classification algorithms, and compact representation through local processing to achieve an acceptable level of performance without sacrificing usability.

The second challenge is capturing user intent, in the form of accurately detecting the instances of the finger touching surface. For example, when we write on a whiteboard with a marker, we get the immediate visual feedback of what is being written, and the physical feedback of the marker touching the whiteboard. As a consequence of the physical feedback, we know exactly when we are writing on the surface versus when we are moving the marker to a new location.

Similar to writing on a whiteboard, when the user's finger interacts with a surface, we need to capture the instances where the finger is actually touching the surface – indicating actual "writing" – versus the finger gliding in the air. While instrumenting a user's fingertip directly can solve this easily, it will affect the usability significantly since we are obstructing the user's natural touch and providing a platform that is too awkward to wear throughout the day. We use a combination of touch induced pressure and motion induced audio sensing to achieve accurate finger touch and motion detection to capture user intent.

A third challenge is to achieve perpetual operation where users do not have to take off the ring and charge it, giving the user an experience similar to wearing a conventional ring. While this is generally a challenge for an active device with a wireless link, this becomes even more challenging due to the limited battery capacity available on a ring platform.

We implemented a ring prototype that addresses these challenges. The ring uses energy efficient finger-tendon based touch detection and audio-based motion detection to capture user interaction instances. A light-weight multi-classifier solution accurately classifies 23 different gesture primitives. The ring harvests energy from an NFC-enabled phone held in a user's hand to achieve perpetual operation of the ring without the need for explicit charging of the battery. With a 10 mAh battery, the ring can support more than 10 hours of active user interactions.

The remainder of the paper is organized as follows. Section 2, gives an overview of the ring platform. Section 3 describes our solutions for detecting surface interactions. Section 4, describes the design of our gesture classification solutions. Section 5 describes the ring prototype implementation. In section 6, we evaluate the gesture classification solution and quantify the energy consumption and harvesting capabilities of the hardware platform. We present relevant related work in Section 7. Finally, we conclude with discussion and future work in Section 8.

2. PLATFORM DESIGN

Our ring-based gesture input solution consists of 3 components: a finger-worn ring worn by the user, a surface for generating various gestures using the ring-worn finger, and a remote device the user is interacting with that consumes and responds to various gestures.

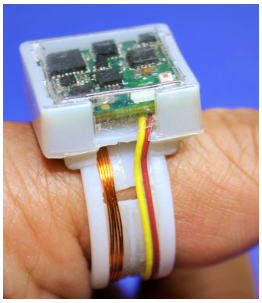

Figure 1: The ring prototype implementation.

The ring is an active device that detects and parses various gestures that the user performs on the surface, and transmits these gestures to a remote device wirelessly. The energy required for these operations are opportunistically harvested from an NFC-enabled phone held in the user's hand, enabling perpetual operation without explicit recharging. Figure 1 shows our current prototype implementation.

The ring is designed to be worn on the index finger of the user, and the user performs various gestures on an available surface using the index finger. The user can interact with almost any type of surface as long as the surface produces sufficient friction, as explained in Section 3. It can identify 23 different gesture primitives for supporting tap, scroll, swipe, and written text-entry on a remote device UI. The identified gestures are transmitted wirelesly to a remote UI device.

Our current prototype implementation using off-the-shelf components and simple manufacturing has a volume which is $\simeq$ 8 times as large as a typical ring in volume (20 mm $\times$ 20 mm surface area, and 5 mm thickness), which can be easily shrunk to the size of a typical ring using production quality manufacturing techniques.

2.1 Design Overview

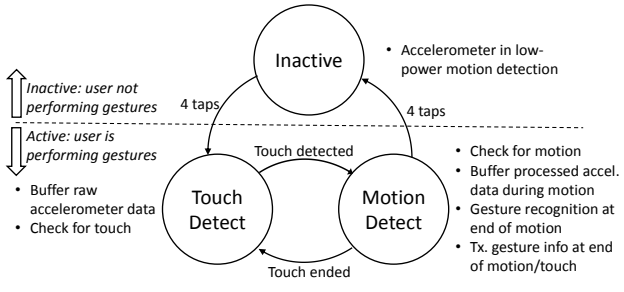

Figure 2: The ring transitions among *active*, *touch-detect*, and *motion-detect* states based on user taps, touch, and motion of the finger on a surface.

The ring has three main sensing elements. An accelerometer is used to capture inertial data during gestures for gesture identification. A force sensitive resistor (FSR), which changes its resistance based on the applied force, is placed next to a tendon of the finger used to detect if the finger is touching a surface. An audio based sensor is used to detect the motion of the finger on the surface, based on the audio generated by the friction between the finger and the surface [21].

During normal operation, the ring is in an *inactive* sleep state to prevent accidental interpretation of day-to-day user activities as gestures. In this state, the ring's accelerometer is in a low-power autonomous motion-detection state.

When the user is ready to enter gestures on an available surface, the ring is brought into active state by tapping the surface 4 times. Once in active state, the ring enters *touch-detect* active state by turning on the touch detection sensor. The 1st tap helps trigger the accelerometer motion-detector, while the rest of the taps are used to reduce accidental triggering of active state. Raw accelerometer readings are collected and buffered during the *touch-detect* state.

When the user touches the surface to enter a gesture, the touch is detected by the touch detector, and the ring enters *motion-detect* state. In the *motion-detect* state, the audio-based motion detector is turned on to detect the motion of the finger along the surface. During this state, processed accelerometer components along plane of the ring are stored. At the end of motion or the touch, these processed accelerometer data is fed to a classifier to identify the gesture.

Once a gesture is identified, this information is transmitted wirelessly to the remote device. Since the user gets immediate feedback on the gesture being received at the remote device, the ring uses a best effort transmission scheme that does not detect or recover from packet loss or collisions.

The gesture classifier on the ring requires the knowledge of the current UI element type when interpreting certain gestures. The ring receives this information from the remote device immediately following the transmission of *tap* gestures which are used for selecting UI elements.

2.2 Gestures

Gesture	Modes	# primitives
Tap	down	1
Swipe	up, down, left, right	4
Scroll	up, down, left, right right-up, left-down	6
Written Text	English characters	12

Table 1: Different gestures, including different modes, and number of classification primitives implemented on the ring.

The ring platform implements the following 4 groups of gestures, which are summarized in Table 1.

Tap. The tap gesture is similar to the typical tap gesture on a touch sensitive surface, or the left mouse click. Similar to touch surfaces and mice, we can use multiple, closely spaced, tap actions to define multiple gestures. We use 2 double-Taps to transition between "active" and "inactive" modes as described above.

Swipe. The swipe gesture is widely popular with touch interfaces for quickly scanning through multiple windows or a collection of items such as photos. The ring platform supports four different swipe actions which include: swipe-up, swipe-down, swipe-left, and swipe-right.

Scroll. The scroll action is also similar to the touch surface or mouse-based scroll. The ring identifies 6 scroll actions: scroll-left, scroll-right, scroll-up, scroll-down, scroll-right-up, and scroll-left-down. The last two diagonal gestures can mimic two-finger based zoom-in and zoom-out on touch displays.

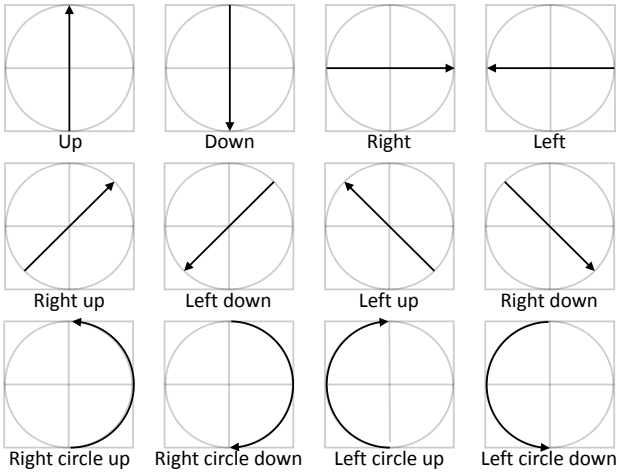

Figure 3: The 12 stroke gesture primitives used for written text entry.

Writing-based text entry We also implement a writing-based English text entry on the ring, where the user simply write the characters on a surface using the index finger.

We use an approach similar to that proposed by Agrawal et al. [1], we treat each character as a combination of multiple primitive shapes called "strokes". Figure 3 shows the 12 strokes used for text entry. The ring identifies these strokes, and also measures inter-stoke latencies which helps identify strikes belonging to a single character. Using techniques similar to those described by Agrawal et al. [1] the stroke identifiers and inter-stroke latencies can be used to identify the individual characters at a remote device.

2.3 Energy Harvesting

The size and weight requirements limit the size of the battery used to power the ring. Based on the size of a typical ring, we selected a 10mAh battery capacity, which can fit in the crown or the band of the ring.

This relatively small 10mAh battery capacity may require the user to recharge the battery several times within a day, which would be a large obstacle to the usability of the platform. To overcome this limitation, we use the *subcarrier based* NFC energy harvesting approach described in [10] to passively recharge the ring battery while the user is holding the phone next to the ring.

To harvest energy from an NFC enabled antenna, the ring has a coil loop wound around the ring band(Figure 1). Winding the coil around the ring band enables us to maximize the size of the loop to achieve better energy harvesting.

3. SURFACE DETECTION

Using an arbitrary, un-instrumented surface for gesture input requires an accurate surface detection mechanism. The benefits of such an approach are twofold: 1) this detection allows the ring to distinguish between gestures made on the surface and spurious motions in the air and 2) provides the user with implicit, natural tactile feedback.

A surface can be detected relatively easily by instrumenting the user's finger tip; however, this is not a practical solution for long-term continuous use due to the impact on the user's touch. Hence, we need to identify and measure indirect physical effects due to touch on a device worn on the user's knuckle. Such indirect measurements require extensive processing to overcome the low signal to noise ratio of the measured signal, making low-power operating challenging.

We use a combination of two techniques to detect the motion of the user's finger on a surface. A pressure sensor embedded in the ring detects the finger touching a surface by sensing the pressure excreted by a *lateral tendon* of the finger, while an energy-efficient audio sensor detects the movement of the finger on the surface by listening to the audio signal generated by the friction between the finger and the surface [21].

3.1 Tendon Pressure-based Touch Detection

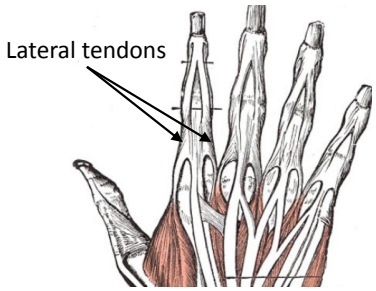

Figure 4: The location of tendons on the back of human hand. The highlighted lateral tendons of the index finger expand outward when the finger touches a surface.

We use the behavior of the lateral tendons of the index finger to detect the finger touching a surface. Figure 4 (copied from *www.wikipedia.org* due the difficulty in accurate reproduction) shows the location of tendons of the human fingers. Of particular interest to us, are the lateral tendons of the fingers.

When a finger touches a surface, these lateral tendons apply a counter force, which causes them to stiffen and move outward laterally. This can be easily observed by gently touching the sides of the index finger between the fist and second knuckle when the finger touches a surface. This lateral motion can be detected using a force sensing resistor (FSR) placed on the inner wall of the ring.

We placed a force sensing resistor between the finger and the inner surface of the ring, and measured the voltage across the sensing resistor.

Figure 5 shows the oscilloscope traces of the measured voltage and the ground truth obtained from a mobile-phone class capacitive touch sensor [4] for both swipe and stroke. We observe that the finger touch can be detected accurately

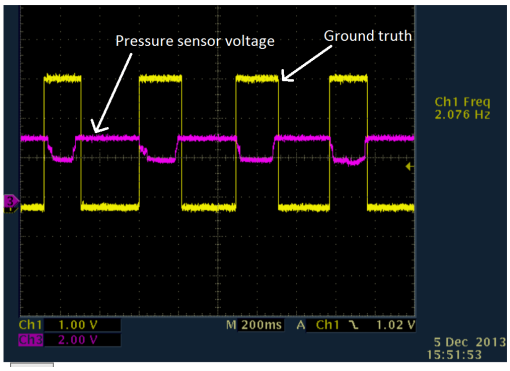

Figure 5: Oscilloscope traces showing the voltage across the force sensing resistor placed between the ring and the finger, and the output from capacitive touch sensor used for ground truth.

by the variation of the measured voltage with a short lag due to the mechanical latency between finger touching the surface and stiffening of the tendon.

We note that a similar stiffening of the tendon happens when the finger is curled. Since a surface interaction session is initiated and terminated by a sequence of taps, the spurious touch detections due to finger curling can be filtered out assuming the user does not curl the finger during the interaction session.

3.2 Audio-based Finger Motion Detection

We use the acoustic signals generated due to the friction when finger moves across the surface to detect finger motion on the surface.

The amount of audio energy emitted from the finger and surface interaction is a function of the speed of finger motion and the physical properties of the surface. The acoustic properties of friction have been well studied in a formal mathematical context [2].

To understand the amount of available acoustic energy on commonly found surfaces, we augmented a glove with a low-power microphone. The microphone is positioned on the index finger's first knuckle, where a ring would typically be worn, with the microphone facing the finger tip. A user performs moderate speed, linear motions across a variety of surfaces in otherwise quiet noise environments. The audio signals picked up by the microphone are recorded by a computer's sound card, while keeping the sound card's gain settings fixed across all experiments. In Figure 6, we show the amount of acoustic energy detected during each experiment normalized by audio length, as well as with respect to each other. As expected, we see rough surfaces, such as wood, cement, or Styrofoam, contain the largest amount of available energy. Smoother surfaces such as paper still emit detectable audio.

Though the acoustic energy of friction is detectable using a low-power microphone, it might be impossible to distinguish the sound of friction from other background noise, it is also important to gain an understanding of the frequency components present in each type of signal. Based on frequency analysis of the audio collected, we find that the frequency components cover a wide band extending from 0 - 15kHz for all the materials tested – this indicates that there are a large

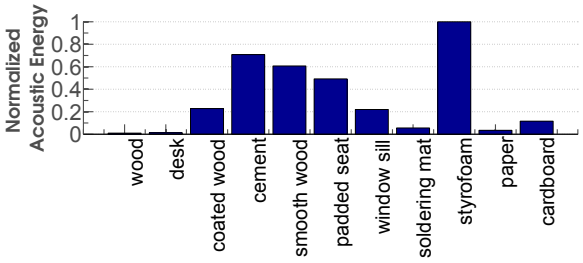

Figure 6: Commonly found materials in office environments produce detectable friction-induced acoustic energy generated by rubbing a finger on their surface.

number of bands that could be selected that do not overlap with other audio signals present.

To get a better sense of how surface friction sounds interact with extrinsic noise, we perform an experiment where we play an audio track on a set of PC speakers while performing periodic finger movements on a sheet of standard office paper.

The audio track is a recording of children playing on a playground with large amounts of talking, yelling, and mechanical noise from playground equipment. We record the audio (signal+noise) observed during the periodic rubbing and plot the resulting spectrogram in Figure 7. The dark region of the plot that extends from 0 - 6 KHz is the artificially-induced playground noise picked up by the microphone. The time-separate dark bands that extend from 7 - 14 KHz each correspond to one individual stroke of the finger moving across the surface. This result indicates that a high frequency, band-pass filter, could be used to filter many of these noise components.

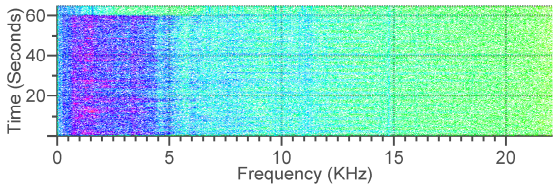

Figure 7: The frequency content of a finger periodically moving across a piece of paper is distinguishable from background noise generated by a PC speaker

To gain a better sense of how bandpass filtering can be used to separate signals from noise, we collect some data from a user entering more realistic gestures and apply a simulated filter to the results – in this experiment, we ask a user to repeatedly input the letter 'L'. After applying a first order bandpass Butterworth filter with corner frequencies at 12 and 14 kHz, we observe the signal plotted in Figure 8. The pair of red bands in the filter frequency range, correspond to individual strokes of the letter 'L'. This indicates that such a filtered audio signal still achieves the necessary time separation required to disambiguate individual strokes of handwriting.

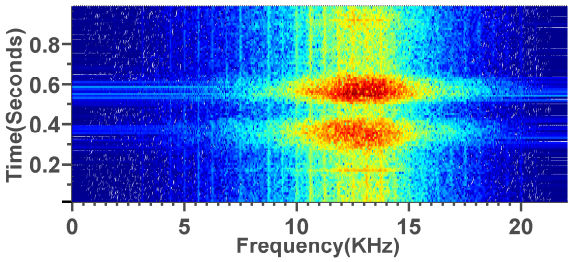

Figure 8: After applying a bandpass filter, the individual strokes of a letter input by a user are easily distinguishable. Here, the pair of red bands denote each line segment of the letter 'L'

Guided by our results we implemented an audio envelope detector with similar filter characteristics, using low-power operational amplifiers for the filtering, gain, and envelope stages.

Using our hardware filter implementation, we log the analog envelope signal using a microcontroller's ADC; additionally, we log the touch state of our touch sensor for ground truth. Both signals are sampled at a rate of 400 Hz. A user performs periodic rubbing gestures across the surface and a tethered PC logs all of the resulting touch and envelope data. In Figure 9, we plot a segment of both of these signals and observe that the audio envelope positions and widths closely align with the touch state of the capacitive touch sensor.

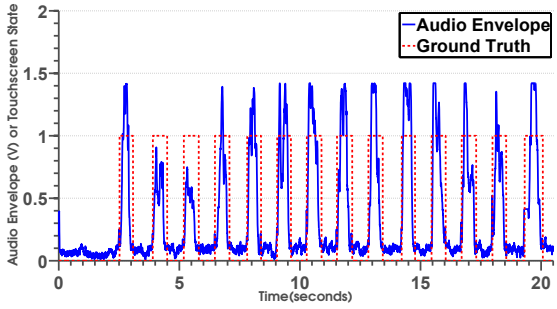

Figure 9: To demonstrate the efficacy of audio envelope-based surface detection, we compare the audio envelope generated by a finger moving along a touchscreen covered with a sheet of office paper. The envelope peaks align closely with the touch state of the capacitive touchscreen.

These results show that audio can be used as a mechanism for surface detection. While audio-based surface detection suffer from intermittent external acoustic noise, the combined pressure and audio based solution can tolerate this noise well.

3.3 Surface detection using Pressure and Audio-based techniques

We use a combination of tendon pressure and audio based techniques for efficient surface detection due to the complementary characteristics of these techniques in terms of power consumption and noise immunity.

The tendon pressure based technique consumes an order of magnitude of power less than the audio-based solution, however, unlike the audio based solution, the pressure-based technique cannot identify finger motion directly. While it might be possible to determine the motion using the accelerometer, the extra processing overhead and segmentation errors introduced can severely affect the system performance.

The audio solution can detect both motion and touch during motion. However, the audio-based solution is affected by ambient audio noise induced errors. While the band-pass filtering removes a significant portion of these errors, we find infrequent short lived errors due to sources such as a high-pitched human voice. The touch detector prevents the accidental triggering of motion sensing due to these errors, while it also prevents cascading failures by touch-based gating of the gesture detection in the presence of noise.

4. GESTURE CLASSIFICATION

As a starting point for designing classifiers for identifying the gesture primitives described in Section 2, we first categorize them based on starting dynamics, real-timeliness, and context dependency of interpretation.

Hard landing gestures start with users finger landing on the surface at a relatively high velocity. Both taps and swipes are hard landing gestures.

Soft landing gestures start with the finger meeting the surface at a low velocity. Both scroll gestures and strokes for text entry are soft landing.

The scroll gestures need **real-time** identification, since the user needs continuous feedback on the current scroll position. Due to short interaction time, both tap and swipe gestures can be identified **non-real-time** after the gesture has completed. The strokes for text entry are also identified **non-real-time**, at the end of each stroke to improve classification accuracy. This is acceptable since the high-level symbols, such as characters, can only be interpreted after collecting all strokes that make up the character, due to lack of absolute position).

Context free gestures are gestures that can be interpreted on the ring without the knowledge of the UI element the user is interacting with. Both tapping and swiping gestures belong to this category.

Context dependent gestures require the knowledge of the current UI element type for correct interpretation by the ring. Some of the stroke and scroll gestures that look identical need to be interpreted differently due to different realtime needs. To enable proper interpretation of these, we assume that the remote device informs the ring when the user starts and stops interacting with a text entry area.

4.1 Resolving Angular Ambiguity of Accelerometer Data

The ring uses the accelerometer data to identify different gestures on a surface. When gestures are performed on a surface, the signature of the gesture is mainly captured in the acceleration components that are parallel to the surface of interaction. Since the ring, when worn on the finger, does not stay parallel to the interaction surface, and since individual users' frame of reference will differ from one another, we need to convert the (X,Y,Z) components of the accelerometer to the (X,Y,Z) components with respect to the interacting plane. In this paper we assume a horizontal plane of interaction, which is true for most of the interaction surfaces available in the environment. We leave the interaction on an arbitrarily inclined plane as future work.

For computing the components along the interacting plane, we use the gravity acceleration vector just prior to movement of the finger to determine the inclination of the finger to the plane, since this angle can vary across users and instances.

To perform this normalization, we need to compute two angles: pitch (θ) and roll (ϕ). To compute these angles, we use well known geometric transformations [7] according to the following equations:

$$\theta_{xyz} = tan^{-1}\left(\frac{-G_{px}}{G_{pz}}\right) \quad (1)$$

$$\phi_{xyz} = tan^{-1}\left(\frac{G_{py}}{\sqrt{G_{px}^2 + G_{pz}^2}}\right) \quad (2)$$

After computing these angles while the finger is stationary, they are applied to subsequent accelerometer samples to compensate for finger orientation while the finger is moving:

$$x_{normal} = -x \cdot cos(-\phi) + z \cdot sin(-\phi) \quad (3)$$

$$y_{normal} = y \cdot cos(\theta) - z \cdot sin(\theta) \quad (4)$$

There are two limitations to correcting for the finger angle in this way. First, we assume that these angles do not change during a gesture – if this assumption is violated, the gravity vector will pollute the X and Y components, and a fraction of the X and Y accelerations will be falsely attributed to the Z axis. Second, we cannot correct for a third orientation angle, yaw(ψ), since we have no gyro (gravity vector provides no information as to how the finger twists in a plane perpendicular to gravity). Thus, we assume that the users finger motion will be perpendicular to their body and the impact of ψ is negligible. If this assumption is violated, the X and Y acceleration components will not be properly separated. Later, we will show that neither of these limitations has a significant impact on classification accuracy.

4.2 Gesture Classification Overview

We use a two-level classification scheme for gesture classification. When the surface detector indicates a touch event, the top-level classifier, called the landing classifier, is invoked to classify the event as either a soft or hard landing event.

Based on the output of the landing classifier and the UI context reported by the end device, one of the low-level classifiers – swipe-tap, stroke, or scroll – is invoked.

The swipe-tap classifier is invoked to handle a hard landing. This classifier uses the surface touch duration – determined by the length of the audio envelope, and the accelerometer samples to classify the event as either a tap gesture or one of the four swipe gestures.

For a soft landing, if the context reports text input, the stroke classifier is invoked. Since there can be multiple consecutive strokes during a single touch, for example when writing the letter `L`, accelerometer data collected after the soft landing is segmented based on the audio envelope. This segmented data is fed to the stroke classifier to classify the stroke to one of the 12 possible strokes. This continues until the finger leaves the surface.

For a soft landing with a non-text input context, the scroll classifier is invoked. The scroll classifier first detects a short *nudge* at the start of the scroll based on the audio envelope, and classifies the data collected between the touch event and the end of the nudge to determine the type of scroll event. After this classification stage, the ring periodically transmits *in-scroll* messages to the end device to provide real-time information on the continued scroll event until the finger stops moving and leaves the surface.

Figure 10 shows a state diagram of how different classification stages are invoked. The rest of this section describes these classification stages in more detail.

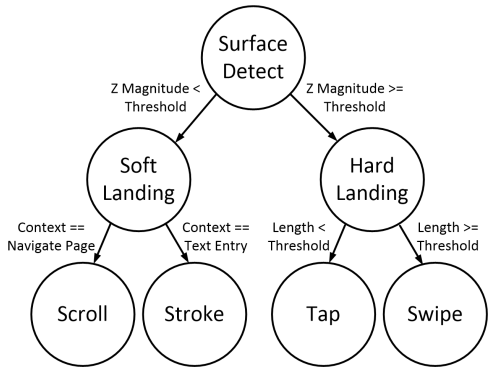

Figure 10: A hard landing has a large magnitude Z-axis component observed at the start of a gesture as compared to the same component of a soft landing.

4.3 Landing classifier

While the finger is moving on a surface, we expect to see negligible activity on an accelerometer's Z-axis. However, when the finger initially lands on the surface, we expect to see the finger abruptly stop, causing large magnitude, short-lived spikes induced by the finger's sudden deceleration. To classify hard landings versus software landings, we exploit this activity on the accelerometer Z-axis at the beginning of a gesture entry. When a surface is initially detected, we look at n samples surrounding the start of motion. These n samples may be used as the input to a heuristic, threshold-based classification or as the input to an SVM classifier. The intuition behind the efficacy of each of these approaches are the existence of large, observable deltas in Z-axis acceleration.

4.4 Swipe-Tap classifier

The swipe-tap classifier is invoked after detecting a hard landing. A tap event is identified by a very short audio envelope length, while a swipe is identified by a longer audio envelope generated as the finger moves across the surface.

A tap gesture requires only the envelope information, while a swipe requires further classification.

4.5 Stroke classifier

To identify strokes, we use an SVM classifier. We use the X, Y, and Z axis accelerometer readings as our feature vector. Since the stroke duration can vary during at different instances as well as across users, we first linearly interpolate 100 points across each x and y axis sample for a given gesture. We then compute a fixed number of averages across these interpolated points for each axis and pass this set of averages to the classifier.

4.6 Scroll classifier

Scroll classification happens after a stroke is detected, based on the context of the remote user interface. A short gesture, called a 'nudge', is detected and classified using an SVM classifier to determine the start of one of the six possible scroll gestures. We mange to accurately classify the scroll action with only a small number of samples since we are using only 6 different scrolls, as opposed to the 12 strokes. After the user performs the nudge, the length of the envelope provides real-time information to the remote device on the progress of the scroll action.

5. NFC ENERGY HARVESTING

A ring platform meant to be worn all the time needs to be of similar size and weight as a typical ring. This requirement limits the size of the battery used to power the ring platform. Based on the size of a typical ring, we assume 10mAh maximum battery capacity.

A small battery may require the user to recharge the battery periodically, which would be a large obstacle to the usability of the platform. To overcome this limitation, we use the `subcarrier based` NFC energy harvesting approach described in [10] to passively recharge the ring battery while the user is holding the phone next to the ring.

To harvest energy from the phone's NFC antenna, the ring has a coil loop wound around the body of the ring as shown in Figure 1. Winding the coil around the ring's body enables us to maximize the size of the loop to achieve better energy harvesting.

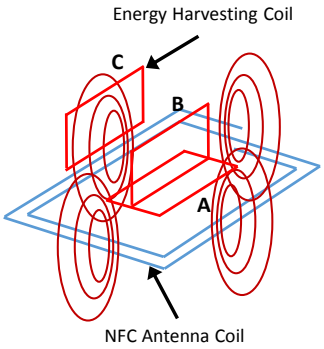

Figure 11: Magnetic field coupling between the energy harvesting coil and the NFC reader coil depends on the relative positioning of coils. Position A has the best magnetic coupling, while B has very little coupling. Position C, representative of the ring, has significant coupling.

Figure 11 illustrates the magnetic field generated by a loop antenna similar to an NFC antenna [19]. We observe that a coil loop placed parallel to the antenna at position **A** has the highest magnetic coupling, hence the best energy scavenging performance. While a loop placed perpendicular to antenna at the center of the antenna (position **B**) has very little flux going through it, a perpendicular loop placed at the edge of

the antenna (position **C**) has some flux going through it due to the bending of the flux around the edge of the coil.

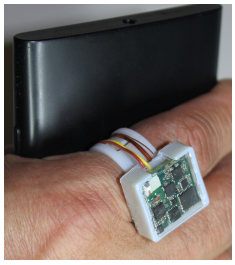

Figure 12: The ring energy harvesting coil is located close to the maximum energy harvesting position when holding a phone.

Figure 12 shows that, when worn on the index finger, the coil of the ring is located close the maximum energy harvesting position (the edge of the NFC coil) of the Nokia Lumia 920 phone, providing significant energy harvesting opportunity.

6. IMPLEMENTATION

6.1 Hardware Platform

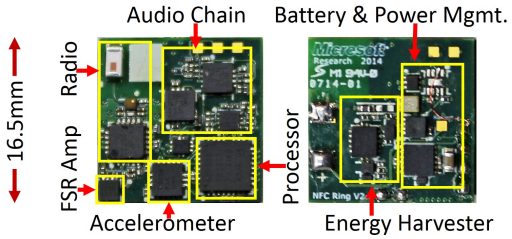

Figure 13: Ring Hardware Modules.

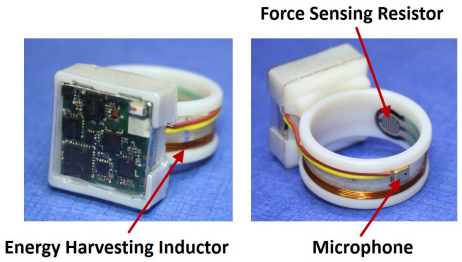

Figure 14: Ring sensing sub modules.

Figure 13 shows various functional sub modules of the Ring circuit board, while figure 14 shows the sensing and energy harvesting peripherals. The circuit board has a MSP430F2274 low-power processor, a CC2500 radio, an ADXL362 accelerometer, audio processing chain, FSR amplifier, a MAX17710 battery manager and power management ICs, and NFC energy harvesting circuitry. The ring has a 10mAh LiPo battery. The energy harvesting coil is wound on the ring band. The FSR and the microphone are also attached to the ring band. The ring PCB and the battery fits within a 16.5mm x 16.5mm x 5mm space.

Figure 15: A bicycle glove, augmented with an accelerometer (left) and MEMS microphone (right) mounted on the first segment of the glove index finger, serves as our data collection platform.

6.2 Surface Detector Implementation

The Surface detector has two pieces: the touch detector and motion detector. The touch detector is implemented using a FSR400 force sensing resistor (FSR) and a LT6003 signal conditioning amplifier. The touch detector can be turned off by power gating.

The audio based motion detector has 4 sub modules: a microphone with a built-in amplifier and high pass filter (SPU0414HR5H-SB), a 4th order high-pass analog filter to filter out typical environmental noise (LT6014), a 40dB amplifier and a low-pass filter (LTC6256), and an envelope detector (LTC6255).

When powered on, the audio stage consumes 670μA, the audio stage can be turned off by power gating.

6.3 SVM Classifier implementation

For gesture classification, we use a multi-class linear-kernel SVM classifier. The classifiers use pair-wise classification, requiring $n(n-1)/$ pair-wise classifers to classify n classes.

Based on our evaluations, we use 4 features for each X and Y axis, resulting in 8 SVM features. Each $(F(i)_X, F(i)_Y)$ feature tuple is calculated by, first breaking all the (x, y) acceleration data in to 4 buckets, and then averaging the data in each bucket. The x and y acceleration component of each data sample is computed immediately after reading the (x, y, z) acceleration data sample from the accelerometer.

Due to limited RAM size, instead of buffering data at 400Hz until the end of a gesture, running averages of each n samples of data is calculated. Value n is selected such that there are at most 60 computed averages for the worst case duration of each gesture class. Once the gesture ends, these precomputed averages are broken into 4 equal-size buckets to compute the 4 feature tuples.

7. EVALUATION

In this section we quantify the ring platform's performance in terms of classification accuracy and rate of energy harvesting and consumption. First, we look at gesture classification accuracy when looking at examples of gesture data (accelerometer and audio) collected from a pool of users. Next, we quantify the energy consumption of the ring and show that we can support the input of many gestures on a single battery charge while being able to quickly replenish the energy buffer using opportunistic NFC-based energy scavenging.

In Section 7.1 all of the data presented is collected using a glove, augmented with acceleration and audio surface de-

tection sensors (depicted in Figure 15). The glove streams 3-axis accelerometer data and and the value of the audio envelope at a rate of 400 Hz. These data samples are stored and collected by a Python script running on a PC; the glove and PC are connected via RS-232. In total, we collected 15 instances of each Swipe and Stroke Gesture from 12 users and 15 instances of each tap and nudge (scroll) gesture from 5 users. Each user's data was collected in a single session, with each type of gesture performed in a group. We also note that the gestures were collected in a relatively quiet noise environment with the gestures performed on a sheet of office paper to mantain a consistent amount of surface friction.

7.1 Gesture Recognition Performance

In this section, we quantify the performance of each classifier component described in Figure 10.

Threshold-based Classifier Evaluation: As described in section 4, the ring needs to be able to distinguish between hard landings and soft landings; after detecting a hard landing, the ring also needs to distinguish between taps and swipes. Since these two types of classification do not rely on time-series accelerometer values, they are likely amenable to a simplistic threshold-based classification algorithm.

To understand how effectively Z-axis magnitude (Hard vs. Soft Landing) or audio envelope width (Tap vs. Swipe) can be used to perform such a classification, we look at the cumulative probability distribution of these features in Figures 16 and 17. The data used to generate these plots was provided by a single user that provided 15 examples of each gesture type. The Z-axis magnitudes were computed as described in Section 4, at the beginning of the audio envelope; the audio envelope width is defined by where the envelope rises and falls above or below 0.2 V. In each of these plots, we observe that most of the probability distribution lies in different halves for each class we attempt to distinguish. For the landing classifier, taps and swipes (hard landings) typically have a much larger Z-axis component than strokes and nudges (soft landings); the CDF indicates that 1.2 G of acceleration might be a good threshold. We make a similar observation for taps versus swipes – taps typically have a much narrower audio envelope than swipes; the CDF indicates that ~ 0.5 seconds might be a good threshold.

Using the thresholds inferred by the corresponding CDF, we construct simple classifers that indicate a hard landing when larger than the Z-acceleration threshold and a soft landing when smaller; a similar classifier was constructed for swipes versus taps using the audio envelope width. We present the classification results in Table 2. We note that these values can be driven even higher with a form factor implementationt that couples more tightly with the finger and user's that have had more time to pratice entering gestures.

Class	Precision	Recall
Soft Landing	93.81%	96.8%
Hard Landing	89.71%	81.33%
Tap	75%	80%
Swipe	94.92%	93.3%

Table 2: Classification thresholds for Envelope Width or Z-axis magnitude may be selected according to a tradeoff between Precision and Recall.

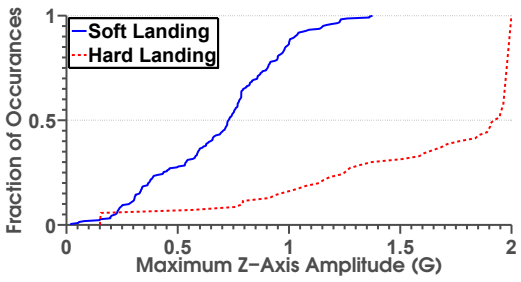

Figure 16: Hard and Soft Landings may be distinguished based on differences in the maximum magnitude of Z-axis acceleration at the beginning of a gesture, as indicated by the audio envelope. This plot shows the cumulative distribution of this magnitude for all 23 gesture classes grouped according to hard versus soft landing types. An acceleration threshold of 1.2 G provides a good tradeoff between precision and recall.

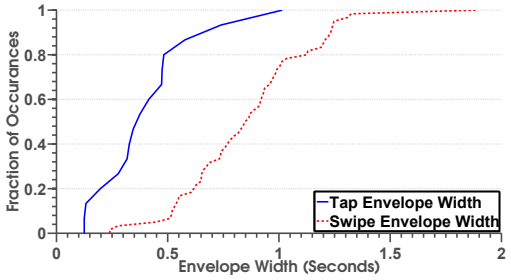

Figure 17: The characterstics of example gestures from one user indicate that most training examples may be correctly classified by using a fixed threshold of 0.51 seconds

After the landing classifer and tap-swipe classifier identify a particular type of gesture, with the exception of the tap gesture, each type is further classified according to time-series accelerometer data segmented by the audio envelope.

Stroke Classifier Evaluation: Unlike the threshold-based classifiers presented previously, strokes require more sophisticated classification techniques. As each gesture is perfomed, the user's finger moves around on a plane defined by the surface – the movements in the X and Y directions can be identified by the X and Y components of acceleration.

There are 3 different types of strokes: orthogonal lines, diagonal lines, and half circles. Orthogonal lines can be identified by looking for acceleration during one half of the gesture, and deceleration during the second half of the gesture – the relative signs of these accelerations indicate the direction, while the dominant axis component defines vertical versus horizontal motions. Diagonal lines are interpreted similarly but with acceleration components in both the X and Y axes. Half circles can be uniquely identified by considering centripetal acceleration, as well as detecting starting and stopping acceleration values with the same sign since the half circle folds back on itself.

One straightforward way to implement a unified classifier for these 3 types of stroke gestures is using a support vector

machine. Guided by the previous intuition, we provide averages of the acceleration during different intervals of each gesture example as described in Section 4. A vector of these features is used to train multi-class libsvm [3] using a linear kernel with the C parameter set to 250.

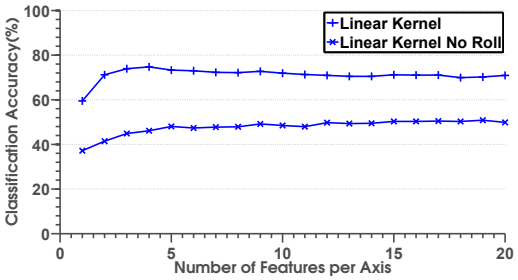

Figure 18: Classification results do not improve significantly beyond 4 features per axis. The metric of evaluation is a per user cross validation across data contributed by 12 users.

Our first evaluation looks at how the trained classifier performs when tested on a new user. The SVM is trained using 11 users' data and tested on the 12th. This process is repeated such that all 12 datasets are cross-validated. In Figure 18 we show the resulting accuracy of this cross-validation. We find that accuracy increases to around 74% as we increase the number of features, saturating with 4 features per axis. To demonstrate the effectiveness of our initial orientation corrections, we plot the classification accuracy when the finger's roll angle is not compensated; we note that the accuracy only reaches around 50%.

Stroke Class	Precision	Recall
Down	61.77%	68.18%
Down Circle Left	82.58%	74.24%
Down Circle Right	80.11%	85.98%
Down Left	58.89%	62.35%
Down Right	60.00%	54.27%
Left	62.36%	63.79%
Right	65.48%	60.44%
Up	67.60%	55.5%
Up Circle Left	86.71%	84.27%
Up Circle Right	86.21%	91.46%
Up Left	68.54%	72.19%
Up Right	53.93%	67.61%

Table 3: Precision and Recall Vary significantly for the 12 Stroke Classes. Circular gestures are easier to classify than diagonals because they are more robust to variations in frame of reference across users.

Next, we look into the confusion matrix for the 12 gesture classes to understand how each class contributes the the aggregate reported accuracy. For each of the 12 swipe classes, we report the precision and recall in Table 3. A key observation here is that the 4 half circle gestures achieve ∼20% better performance than the line segments. Since these 4 gestures have a signature different from the line segments, and unique signatures defined by centripetal acceleration, they are more easily disambiguated. The key challenge in line segments is that untrained users with no feedback from a user interface have difficulty accurately defining the angles required to disambiguate diagonal versus orthogonal lines. More training data and feedback from the user interface could certainly improve this performance, but these improvements are beyond the scope of this work.

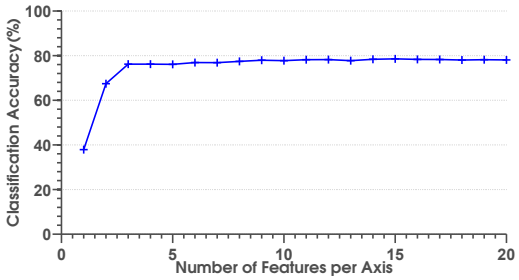

Figure 19: Accuracy results obtained from a ten-fold cross validation applied across all user data indicate that higher accuracy numbers can be achieved given more available training data similar to new users.

To get a better sense of how more training data could potentially help classification, we look at another metric of evaluation. In Figure 19 we look at the accuracy obtained by a 10-fold cross validation that treats all user data uniformly. The user data is randomly partioned into 10 folds; 9 folds are used for training and the remaining fold for testing. These folds are then validated against each other. We note that accuracy improves to around 80%, with accuracy saturating at 10 features per axis.

Swipe Classifier Evaluation: We evaluate the 4 swipe classes using the same approach as strokes. Strokes differ from swipes in two ways: 1) there are fewer swipe classes than stroke classes, and 2) they are typically much shorter in length since they are performed quickly. We peform per user cross-validation and plot the resulting accuracy in Figure 20. We observe around 77% classification accuracy for 4 features per axis. This is because swipes are easier to disambiguate from one another since they are fewer in number and spaced further apart geometrically.

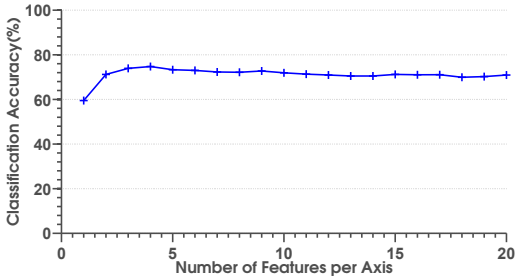

Figure 20: A classifer trained on the swipe classes achieves slightly higher accuracy than the stroke classifier. While swipes typically contain fewer raw samples than strokes, the overall accuracy is higher since there are fewer classes. The test metric is per user cross validation for 12 users.

Nudge Classifier Evaluation: Finally, we present accuracy results for nudge features in Figure 21. The metric of evalution is per user cross-validation and again, note that accuracy saturates at 68% for 4 features per axis. This accuracy is the lowest since each gesture example consists of only a few raw samples. Other sources of error are a result of the diagonal line segments use to implement the "zoom" type gesture. Finally, we note that performance is also diminished as we only currently have training examples of nudges for 5 users as opposed to the 12 used for strokes and swipes.

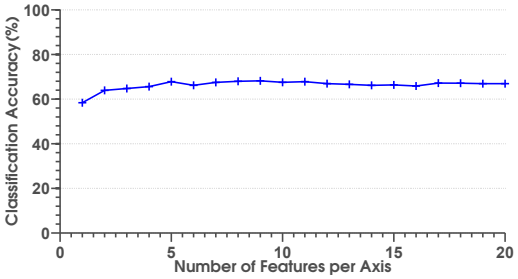

Figure 21: A classifer trained on the nudge classes achieves around 70% accuracy with 4 features of data per X/Y axis. While there are fewer nudge classes and swipes, the small number of samples available makes classification difficult.

Sensitivity to Angular Variations: While we have shown that classification for our collected training and testing data, one question remains: How well does classification perform when there are errors in angular correction for new data with respect to the trained model? Since the classifier is trained with one frame of reference in mind, performance will likely suffer if this frame of reference changes after training (i.e. finger angles are miscalculated). This is a distinct possibility given varying noise environments and differences in surface material properties. To understand how well classification performs in spite of these errors, we train the SVM classifier assuming that angles are computed correctly using our the described techniques, but then perform per user cross-validation using test data with an artificially induced pitch error, applied to all testing examples for all users. We use the a linear kernel with 4 average features per axis and repeatedly run cross validation for different amounts of error. We plot the results in Figure 22. We note that the classification accuracy only diminishes by 2% for angles as large as +/- 3 degrees.

7.2 Power Consumption

To determine the average power consumption and battery life, we assume a workload with 2Hz gesture frequency. We also assume that users spend equal amounts of time touching the the surface and hovering above the surface, resulting in a 50% duty cycle (250ms execution time) for touch detect and motion detect.

Gestures are encoded in 10 byte RF messages. The messages are transmitted using best effort with no collision avoidance or detection. Immediately following the messages with *tap* gestures, the ring expects to receive a 10byte message encoding the current UI element type for context-based gesture classification. We assume that 25% of all the gestures are tap gestures.

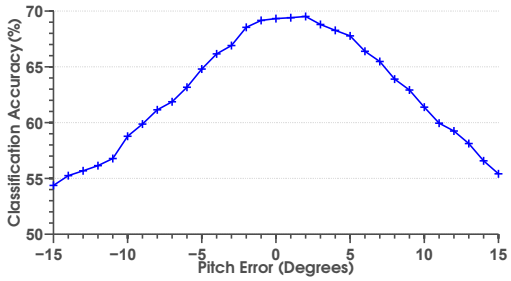

Figure 22: The model learned by the SVM classifer is relatively immune to errors in gesture angle computation. The classifier maintains high accuracy with pitch angle errors consisting of a few degrees in either direction.

The hardware modules on the ring are powered from the battery through low-power linear regulators with negligible leakage current. Hence, the total current drawn from the battery is the sum of currents consumed by individual HW modules.

Since the battery capacity is given in mAh, we use the average current consumption as the measure of power consumption. This enables a direct comparison of how the power consumption of different modules impact the overall battery lifetime. While the active current consumption of hardware modules were measured, the sleep currents were estimated from device data sheets.

The following is a breakdown of current consumption of the different system modules under different operating conditions:

- The MSP430F2274 microcontroller consumes 450 μA at 1MHz clock. It operates at a 30% duty cycle during *touch-detect*, resulting in 135 μA average current.
- During a touch detect, the pressure sensor-based touch detector consumes 11 μA, while the audio-based motion detector is turned off with negligible current consumption. During motion detect, the touch detector and motion detector consume 30 μA and 670 μA respectively.
- After waking up from deep sleep, the CC2500 radio takes 5.52 ms for register configuration and message transmission for a 10 byte message. At -30dBm transmit power, the measured average current consumption within the 5.52 ms interval is 4.32 mA. At 2 Hz gesture frequency, this translates to an average current of 47.7 μA per gesture (neglecting the 0.4 μA sleep current).
- During bi-directional communication, the remote device transmits a 10 byte message immediately following the receipt of a *tap* gesture message. The CC2500 radio takes 8.6 ms for register configuration, message transmission, and reception. The measured average current consumption during this interval is 9.85 mA. This translates to 169 μA average current per gesture.
- Assuming 25% of the gestures are *tap* gestures that require a response from the remote device, the average current consumption of CC2500 per gesture is 78.0 μA.

Module	Touch detect (μA)	Motion detect (μA)	Inactive (μA)
Microcontroller	135	450	0.1
Touch detector	11	30	$\simeq$0
Motion detector	$\simeq$0	670	$\simeq$0
Accelerometer	5	5	0.27
Radio	78		0.4
Total	229	1233	0.77

Table 4: The current consumption of sub-modules of the ring during *touch-detect*, *motion-detect*, and *inactive* states.

Table 4 summarizes the current consumption of modules during touch detect, touch process, and inactive states. Under the given workload, the ring consumes only 731μA when active on average. It consumes only 0.8μA when inactive. With a fully charged 10mAh battery, the ring achieves more than 10 hours of active interaction time.

7.3 Energy Harvesting Performance

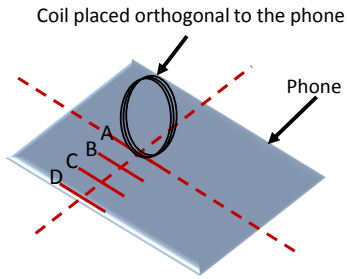

Figure 23: Energy harvested with a tuned coil held perpendicular to the surface of the phone were measured at 4 different locations on the phone.

We first determine the maximum power that can be harvested from the phone using a tuned harvesting coil with a matched resistive load. To measure this, we placed the tuned coil on the back of an Lumia 920 phone, orthogonal to the surface of the phone, at 4 equally spaced locations labeled (**A**,**B**,**C**,**D**) as shown in Figure 23. We used a 1.5 cm diameter coil with 10 turns.

Figure 24 shows the power that can be harvested at these locations. We observe that the ring can harvest up to 18 mW of power when placed at C, which is right next to where a ring worn on the index finger is typically located.

We next measured the actual power harvested by the ring when charging the built in battery by attaching a resistor between with the battery -ve terminal and the ground. We used a 10 ohm 0.1% resistor for measuring the charging current. We measured a 3.27mA charging current when the battery voltage is lower than the charger cut off voltage. The trickle charge current when the battery voltage is above the charger cut off was negligible.

7.4 Trace-driven Performance Evaluation

To further validate our harvesting claims, we look at a trace driven evaluation of ring harvesting performance

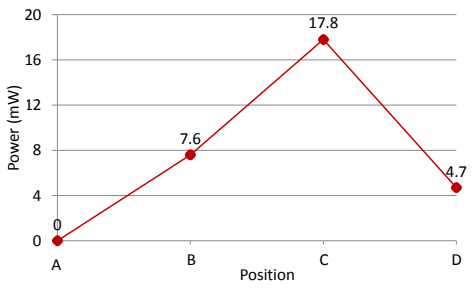

Figure 24: The maximum power harvested by a tuned harvesting coil placed at different locations of a Lumia 920 phone.

across week long intervals of normal phone usage. To understand what normal phone usage looks at, we use a set of traces provided by the LiveLab project at Rice University [18] – these traces give detailed useage logs of 34 iPhone users over the span of one year. Of particular interest to this paper are the intervals during which the phone's screen is unlocked, as these are opportunities to harvest energy from the phone's NFC interface.

Our simulation study assumes that while a user is actively using their phone with the screen unlocked, the ring-worn finger is positioned favorably on the back of the phone, allowing the ring to harvest power according to position B in Figure 24 and assume the harvesting rate measured by the 10 ohm resistor reported in §7.3. While harvesting this power, the ring is able to store energy in its battery equal to the difference of this harvesting rate and the sleep power consumption shown in Table 4. After the battery reaches 96% of its total capacity, the battery is trickle charged – in practice we found the trickle charge current to be negligible, thus our simulation does not consider the final 4% of the battery's capacity. We also assume that the initial 5% of the battery capacity is unavailable, as a protection circuit prevents the battery from going below 2.5 V to prevent damage caused by deep discharge.

The simulation workloads we consider, consist of periodic intervals of gesture inputs that occur once every 2 hours and last for a variable number of minutes. We do not consider intervals that overlap with a screen unlock event, as the user is interacting with their phone rather than a remote device via the ring. During a gesture input session, the ring spends equal amounts of time in power states according to *touch* and *motion detect*, again according to Table 4. This implies that the finger spends the same amount of time in the air and performing gestures on the surface. During each of these gesture input sessions, we consider the amount of energy at the start of each session and consider the ring is "available" if the battery is charged beyond 5% of its total capacity, plus the energy required for the current gesture interval duration. The ring is considered unavailable for gesture input if below this threshold. In Figure 25, we show the fraction of occurances during which the ring is available. Each box plot is computed over all 34 users, for gesture input durations ranging from 1 - 30 minutes. Even under very aggressive workloads where the ring is active for 20 minutes out of every 2 hours, the median of ring availability computed across all 31 users is still around 90%. We also note that these

performance numbers are considered a worst-case, since in a real-world scenario, it would be very uncommon for users to perform gesture inputs during the middle of the night.

We further validate the worst case nature of these results by look into the timing characterstics of time periods where the ring was not available. In Figure 26, we plot a CDF showing the amount of time between the last charging opportunity and instance of the ring being unavailable due to energy starvation. We note that even for the 30 minute workload, 75% of failures occur after 4 hours of mobile phone inactivity. These failures are likely induced by the previousy mentioned "night effect" and other longer periods of inactivity. More realistic workloads would likely occur more closely to periods of phone inactivity while people are active and engaged in other activities.

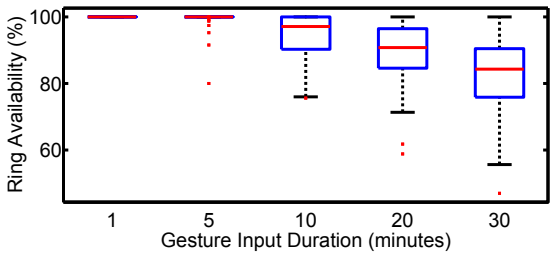

Figure 25: A trace-driven simulation of power harvested opportunistically by the ring platform indicates more than 80% availability for 31 different users over the span of a year. A periodic useage model is considered, where the ring is usd for gesture input for the length specified every 2 hours.

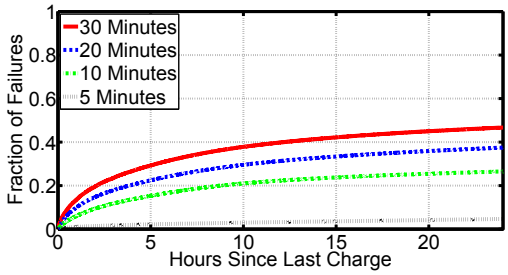

Figure 26: A CDF of the time between last charge interval and gesture failure shows that most failures occur as a a result of energy starvation after long periods of phone activity. In real world use cases, gesture input sessions would likely occur within a couple of hours of phone activity.

8. RELATED WORK

The use of accelerometers for gesture recognition has been studied extensively – we provide a summary of the most relevant related work in Table 5. These systems primarily look at performing gestures in the air using commodity devices such as a mobile phone or the Nintendo Wii Remote – while our work focused on the system power consumption and perpetual availability possible in a wearable ring form factor devices, these systems instead focused on techniques that maximize classification accuracy. The Nintendo Wiimote was the first widely popular device that used acceleration-based gestures as a primary input mechanism; Schlomer et al. [17] show it was possible to disambiguate a small gesture vocabulary using an HMM-based learning algorithm using the output of the Wii Remote. The uWave system [13] uses a dynamic time warping based technique to align gesture inputs to templates that denote different gestures. Agrawal et al. [1] describe a mobile phone based system for combining indivdual strokes into characters. The PhonePoint system is similar to our work in that it uses stroke primitives to build up more complex characters and words. We view our work as complementary, as we focused primarily on the energy efficient detection of individual strokes.

System Name	Accuracy	Battery Life
Wii Remote [17]	85 - 95%	60 hours
uWave [13]	91 - 99%	60 hours
PhonePoint [1]	46 - 100%	40 hours

Table 5: A number of related gesture-based input solutions have been proposed. None of them simultaneously offer continuous system availability within a self-contained, wearable form factor.

Also similar to our work are other wearable ring-like platforms. Magic Finger [20] is a device worn on the finger tip and uses a low-resolution camera and motion sensor to track movement on a surface. Unlike our work, Magic Finger focuses on UI and does not consider system power consumption. The iRing [16] senses changes in finger position using infrared sensors looking inward at the wearer's skin. Like our platform, the ring platform presented by Zhang et al [21] also uses audio and acceleration sensors to track movements on a surface. This iRing performs audio signal processing on sound conducted through the wearer's bone; in contrast, our ring platform uses the envelope of audio emitted from the surface rather than through the finger and uses an accelerometer to track the direction of motion. We also find that existing commercial ring-based UI controls are currently too bulky to be continuously wearable [8].

Finally, there are other systems that leverage the acoustics of friction [2] to interact with different types of surfaces. Stane [15] consists of a handheld device that used scratching and tapping sounds as gesture inputs to a music player. Several systems [11, 14, 12] instrument the surface itself, rather than the finger and detect the noise produced by user interactions. The SurfaceLink system [9] allows multiple devices to communicate with each other and perform gestures via vibrations and sound travelling along a shared surface. Instead of using a surface directly as a communication channel, our work uses the surface to partition accelerometer data and provide the user with impicit tactile feedback that aids the user in gesture entry tasks.

9. DISCUSSION

This paper focused on the design aspects of an energy-constrained, wearable ring platform. In the future, our system design can be enhanced to tackle other important challenges such as improving classification accuracy, improving noise resilience of audio-based motion detection, and better quantifying the system performance on a larger variety of surfaces.

Classification Accuracy: The gesture classification performance of our system can be improved in several ways. First, in the evaluation presented in this paper, users were not given a frame of reference for gesture entry, likely causing more noise in training data. Another major drawback was the lack of feedback given to a user after performing each gesture input. Adding this feedback would greatly enhance the ability of users to learn how to correctly enter gesture examples. Finally, since machine learning-based classification methods are dependent on the availability of large amounts of training data, our system could greatly benefif from an increase in the number of users and training examples used to build the SVM classifier models. Based on results obtained from self cross-validation, we expect additional training data to increase the overall accuracy by at least 10%.

Noise Resilience: We observed that audio-based surface detection suffers due to high frequency noise, such as noise from ultrasonic room occupancy sensors. This could be mitigated with an audio channel selecting bandpass filter, as well as other time-domain filtering techniques. After adding these techniques, an evaluation would quantify surface detection accuracy in a variety of expected noise envionments.

Performance of Different Surfaces: Our classification results were based on audio data collected from common office paper. We note that in Figure 6, office paper has little acoustic surface friction energy compared to other types of materials we benchmarked making our initial classification results encouraging. However, a more extensive evaluation would look more closely at other commonly available surfaces such as coffee tables and desks; we also note that the type of surface will also have a major impact on its resilience to extrinsic noise.

10. CONCLUSION

This paper presented a low-power wearable ring platform that enables users to enter gestures by interacting on arbitrary surfaces. The ring uses energy efficient finger-tendon based touch detection and audio-based motion detection to capture user interaction instances. A light-weight multi-classifier solution accurately classifies 23 different gesture primitives. Using a 10 mAh battery, charged by energy harvested from an NFC-enabled phone, the ring can support more than 10 hours of active user interactions.

11. ACKNOWLEDGEMENTS

This research was partially funded by NSF grants CNS-1218586, CNS-1217606, and CNS-1239341. We thank our shepherd Prabal Dutta for providing guidance in preparing our final draft and the anonymous reviewers for their insightful comments. We also thank the anonymous volunteers that provided examples of gesture inputs.

12. REFERENCES

[1] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter. Using mobile phones to write in air. In *MobiSys '11*, pages 15–28. ACM.

[2] A. Akay. Acoustics of friction. *The Journal of the Acoustical Society of America*, 111:1525, 2002.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. volume 2, pages 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] Cypress Semiconductor. CY3290-TMA400EVK TrueTouch Gen4 Evaluation Test Kit (`http://www.cypress.com/?rID=58447`), June 2013.

[5] M. Dunlop and S. Brewster. The challenge of mobile devices for human computer interaction. *Personal Ubiquitous Computing*, 6(4):235–236, Jan. 2002.

[6] L. Eronen and P. Vuorimaa. User interfaces for digital television: A navigator case study. In *AVI '00*, pages 276–279. ACM.

[7] Freescale Semiconductor. Tilt Sensing Using a Three-axis Accelerometer (`http://www.freescale.com/files/sensors/doc/app\_note/AN3461.pdf`).

[8] Genius KYE Systems Group. Wireless Thumb Cursor Controller (`http://www.geniusnet.com/wSite/ct?xItem=47711&ctNode=105`), 2011.

[9] M. Goel, B. Lee, M. T. I. Aumi, S. Patel, G. Borriello, S. Hibino, and J. Begole. Surfacelink: Using inertial and acoustic sensing to enable multi-device interaction on a surface. In *CHI '14*.

[10] J. Gummeson, B. Priyantha, D. Ganesan, D. Thrasher, and P. Zhang. EnGarde: protecting the mobile phone from malicious NFC interactions. In *MobiSys '13*, pages 445–458.

[11] C. Harrison and S. E. Hudson. Scratch input: creating large, inexpensive, unpowered and mobile finger input surfaces. In *UIST '08*, pages 205–208. ACM.

[12] C. Harrison, J. Schwarz, and S. E. Hudson. Tapsense: enhancing finger interaction on touch surfaces. In *UIST '11*, pages 627–636. ACM.

[13] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[14] P. Lopes, R. Jota, and J. A. Jorge. Augmenting touch interaction through acoustic sensing. In *ITS '11*, pages 53–56. ACM.

[15] R. Murray-Smith, J. Williamson, S. Hughes, and T. Quaade. Stane: synthesized surfaces for tactile input. In *CHI '08*, pages 1299–1302. ACM.

[16] M. Ogata, Y. Sugiura, H. Osawa, and M. Imai. iring: intelligent ring using infrared reflection. In *UIST '12*, pages 131–136. ACM.

[17] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a Wii controller. In *TEI '08*, pages 11–14. ACM.

[18] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[19] Texas Instruments. HF Antenna Design Notes (http://www.ti.com/lit/an/scba034/scba034.pdf).

[20] X.-D. Yang, T. Grossman, D. Wigdor, and G. Fitzmaurice. Magic finger: Always-available input through finger instrumentation. In *UIST '12*, pages 147–156. ACM.

[21] B. Zhang, Y. Chen, Y. Qian, and X. Wang. A ring-shaped interactive device for large remote display and mobile device control. In *UbiComp '11*, pages 473–474. ACM, 2011.