SINGLE MICROPHONE TAP LOCALIZATION

by

Tusi Chowdhury

A thesis submitted in conformity with the requirements
for the degree of Masters
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2013 by Tusi Chowdhury
This thesis explores a single microphone tap localization interface for smartphones - Extended Touch(ET), that detects user-tapped locations on any neighboring surface. The algorithm combines accelerometer and microphone detection making it robust to noise, and does not require knowledge of surface parameters or sensor positioning. It uses acoustic signal as the feature vector and solves for tap inference in two phases - training and detection. The training phase builds a prior-model of the system by storing one or more templates of known tap locations. These templates are used in the detection phase to carry out a k-nearest neighbor classification to detect new tap locations. The algorithm achieves a 92% detection rate on knock taps. A method to detect contiguous tap locations is also proposed.
Dedication

I dedicate my MASC thesis to my dear parents - Pally Chowdhury and Sajal Chowdhury, for the advice, guidance, and opportunities they have provided me throughout my personal and professional life.

Acknowledgements

I would like to say special thanks to my Supervisor Dr. Parham Aarabi for his mentorship, guidance, and for giving me the opportunity to work with him on Extended Touch for my MASC thesis.
I would also like to thank my colleagues - the extended touch team - Keith, Max, Kai for helping with data collection, iPhone application development and implementation.
## Contents

1 Introduction .................................................. 1  
   1.1 Motivation ................................................. 1  
   1.2 Objective ................................................ 3  

2 Background .................................................. 5  
   2.1 Prior Research ............................................. 5  
   2.2 Acoustic Wave Propagation And Analysis Techniques ............... 7  
      2.2.1 Time delay of arrival ................................. 8  
      2.2.2 The image model ...................................... 11  
      2.2.3 Cross-Correlation ................................. 13  
   2.3 Machine Learning Techniques ................................ 13  
      2.3.1 K Nearest Neighbor ................................. 13  
      2.3.2 Regression Analysis ................................. 14  

3 Tap Detection and Classification ................................ 15  
   3.1 Sensors and Tap Detection .................................. 15  
      3.1.1 Microphone ........................................... 15  
      3.1.2 Accelerometer ....................................... 19  
      3.1.3 Gyroscope ............................................ 22  
   3.2 Tap Classification - Binary Classifier ......................... 24  
      3.2.1 Nearest Neighbor Classifier ....................... 24  
      3.2.2 Logistic Regression ................................. 28  
   3.3 Multi-class Classification, Training and Test Results ............ 30  
   3.4 Methods for combining Multiple Templates and Results ............. 33  
      3.4.1 Inconsistent tap ..................................... 33  
      3.4.2 Speed Boost Using Centroid of Templates ................ 35  
      3.4.3 Performance Evaluation ............................. 37
4 Sensor Fusion for a Robust Classifier 39
  4.1 Tap detection in the presence of background sound 39
  4.2 Misclassification rate with degrading SNR 44

5 Contiguous Tap Detection 49
  5.1 Inference in 1D 50
    5.1.1 Preliminary Results 50
    5.1.2 iPhone implementation and Results 51
  5.2 Linear Regression - Training and Results 52
  5.3 Future Research - Ensemble Method 58

6 Conclusion 60
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Glass table</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Hollow wood table</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Metal sheet</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Aggregated Results from three surfaces - glass, wood, metal, 8 configurations of pairwise taps</td>
<td>27</td>
</tr>
<tr>
<td>3.5</td>
<td>Classification of 17 locations, 1 template/location</td>
<td>31</td>
</tr>
<tr>
<td>3.6</td>
<td>Misclassification of Soft Taps</td>
<td>31</td>
</tr>
<tr>
<td>3.7</td>
<td>Mixture of Taps, 1 training/(x,y)/type</td>
<td>32</td>
</tr>
<tr>
<td>3.8</td>
<td>Mixture of Taps, 1 training/(x,y)</td>
<td>32</td>
</tr>
<tr>
<td>3.9</td>
<td>Classification Success Rate for Different Models</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Confusion matrix, talking dataset 1, threshold 0.1, 325 samples: 3 taps locations, 0.1s sliding window</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Confusion matrix, talking dataset 2, 7 tap locations, threshold 0.05, 0.1s sliding window</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Confusion matrix, talking dataset 2, 7 tap locations, threshold 0.05, 0.1s sliding window (continued)</td>
<td>43</td>
</tr>
<tr>
<td>4.4</td>
<td>Confusion matrix for dataset 1, tier classifier using sensor fusion, 3 tap locations, 0.1s detection window</td>
<td>43</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Inference of user-tapped location on the surface .............................................. 4

2.1 Time delay of arrival from source (x, y) to the sensors .......................... 8
2.2 Gyroscope detection for knock tap at (-10,0) on glass ................................ 9
2.3 Accelerometer detection for knock tap at (-10,0) on glass ................... 10
2.4 Microphone detection for knock tap at (-10,0) on glass ........................ 11

3.1 Microphone detection of tap impulse response for various tap types ... 17
3.2 Max Cross-Correlation of sample taps with different template taps. Microphone detection shows taps at the same location are highly correlated. 18
3.3 Accelerometer detection in x and y axes for soft tap (10,0) .................... 19
3.4 Accelerometer detection in x and y axes for knock tap at (-10,0) ........ 20
3.5 Accelerometer z-axis detection ................................................................. 20
3.6 maximum cross-correlation of knock tap at (20,0) with template taps at various locations ................................................................. 21
3.7 Gyroscope detection for knock tap at (-10,0) on glass ........................ 23
3.8 Different surfaces used to test classification on iPhone ......................... 25
3.9 Tap location pairs for live classification test on iPhone ........................ 26
3.10 Training and test error rates for logistic regression binary classifier ...... 29
3.11 Classification Rate vs. Number of Templates per Location .................. 34
3.12 Two taps at the same location without alignment ............................... 36

4.1 Examples of tap detection with talking in the background at location (0,20) 40
4.2 Examples of tap detection with talking in the background at location (-10,0) 41
4.3 Error rate, threshold = 0.1, 0.1s moving window across all test samples, a total of 325 detection windows ................................................................. 42
4.4 Threshold = 0.1 ................................................................................. 45
4.5 Misclassification rate as function of degrading SNR with talking in the background with threshold set to 0.1, for tap locations (0,20), (10,0) and (-10,0) ................................................................. 45
4.6 Threshold = 0.05 ........................................................................ 46
4.7 Misclassification rate as function of degrading SNR with talking in the background with threshold set to 0.05, for tap locations (0,20), (10,0) and (-10,0) ................................................................. 46
4.8 Misclassification rate as function of degrading SNR with talking in the background, threshold = 0.05, locations (+10, 0), (+10,+10), (+15, 0), (+20, 0), (+20,+20), (+30,+30), (+5, 0) ................................................................. 47

5.1 Maximum cross-correlation value drops as the distance increases between the taps that are being compared .............................................. 50
5.2 Maximum cross-correlation between training taps on x-axis .............. 53
5.3 Distance prediction error as function of distance to the template taps for test taps at (-10,0) and (-15, 0) ....................................................... 54
5.4 Distance prediction error as function of distance to the template tap for test tap at (-5, 0) ................................................................. 55
5.5 Distance prediction error as function of distance to the template tap for test tap at (20, 0) ................................................................. 55
5.6 Error distribution seen among the test templates for 0, and 5 cm away from the template taps ....................................................... 56
5.7 Errors seen among the test templates for 10 cm, and 15 cm away from the template taps ....................................................... 57
Chapter 1

Introduction

1.1 Motivation

Transforming ordinary surfaces into an input medium for digital devices enables inexpensive and convenient alternatives to modern-day touch interfaces. The concept behind extended touch is to develop such a technology where once a portable device is placed on any surface, the entire surface becomes tap sensitive. Modern day smartphones have various built-in sensors with capabilities to understand and respond to their surroundings. However, the current method of interacting with the phone through the touch-screen could at times be limiting. If it was possible to extend the screen real-estate to the table, the counter or any surface the phone is placed on, it could lead to a more natural way of interacting with the device. The goal behind extended touch technology is to develop such a tap detection system that can detect user-taps on any neighboring surface. A solution that results in high-accuracy detection and adapts to any surface, could be used to build many new human computer interaction platforms including the ones outlined below:

Better Gaming Platform

Extended touch will enable a larger surface through which one can control their game movements making it easier to interact and play using their smartphones. One example application could be if a user is playing a racing game, he or she can map four different tap locations to the actions of acceleration, break, turning left and right. Moreover, extended-touch capability could enable an easier way to use a smartphone to play any game with multiple players. One example of such a multiplayer extension could be a ping pong game played using one device, where the screen is projected onto a television or
projector through Airplay. When the phone is placed on a table, using extended touch, tap locations close to different players can be mapped to locations on the game’s ping-pong table. The players then tap on the table near them to hit the ball. A third example of gaming application could be a virtual poker manager or dealer using extended touch. The device could be placed anywhere on the table, and each player makes a bet by tapping at a location close to them. An application can be built to track each tap location per player, and manage the bets and progression of the poker game for all the players. Extended touch creates the possibility of developing many such creative applications just by extending the area through which we can control the device.

Virtual Musical Instruments

Similar to the gaming application, having a larger tactile surface area as the input-interface to portable devices, one could development various virtual musical instruments applications that can provide a more natural user experience. One example of a virtual instrument using extended touch is a Piano application where seven different tap locations on a neighboring table could be registered as different musical notes. The user can tap one of these notes (tap locations defined by the user) to simulate playing the octave keys. The detection could be extended to include more keys to develop a virtual keyboard the user can play anywhere by just tapping on the surface their smartphone is placed on. One could also imagine many similar applications where more instruments could be added, such as drums, with addition of multiple players who can play the instruments simultaneously on the same surface. We developed such a keyboard using the extended touch technology whose demo can be found at [1].

Page Navigation from a distance

Extended touch can be applied to develop applications that greatly simplifies the task of presenting a PDF document, or Keynote presentations from a smartphone or tablet. If the presenter places the device on a table, which is being used for streaming the presentation, the entire table can become an input interface using extended touch by mapping different tap locations to browsing the presentation document. Therefore, the presenter or anyone else sitting around the table could move forward or backward in the document by just tapping on the table near them which can be detected and used to facilitate navigation through the pages. One such demo application was developed and the video demo could be found at [1].

There are many other scenarios such as while the user is cooking or cleaning, where
extended-touch applied to browsing or page-navigation can be extremely useful. One such example is a cooking application where it is not desirable to touch your iPhone or iPad’s touch screen with your food-covered hands. Thus looking at the next page or next instruction by tapping near the phone would lead to a better user experience while keeping the iPhone or iPad handsfree.

**Multiple user defined activities launcher**

Extended touch can be integrated into the operating system of any portable device where the user can define their own actions corresponding to different tap locations. This will allow developers to write applications where the user can snooze alarm clocks, send emails or texts, receive or initiate phone calls by tapping on different locations on the neighboring surface of the smartphone.

**Workout assistance/monitor**

Extended touch technology can be used to detect not just hand taps, but also foot step locations, dance taps, or jump locations on the floor. One can use this detection to develop a work-out instruction application, similar to that of Dance Dance Revolution by Konami, but without the use of any external sensor-loaded mat, and making use of only the built-in sensors available in smartphones. A such application was developed to test out extended touch. It was found that depending on the floor type the performance may vary. For example, a concrete floor generates less acoustic vibration compared to a wooden floor. However, extended touch can still be used to obtain tap detection with lower accuracy rate than the table-type surfaces.

Therefore, extended touch technology can potentially convert an every day surface into a keyboard, touch-input, gaming controller, gaming platform, musical instrument simulator, or any types of intuitive interface. This technology could potentially be applied to infer key-strokes from a physical keyboard that is placed on the same surface, thus introducing a new topic in mobile security exploration. Hence, the goal of this thesis to explore and build an algorithm to implement extended touch capability using sensors available on iOS platform.

**1.2 Objective**

The objective of this thesis is to investigate and develop such a touch-based user interface for iOS platform that can detect, classify and infer user-tapped locations on any
neighboring surface with reasonable accuracy. The problem we are aiming to solve is demonstrated in Figure 1.1. Our objective is to develop and employ a learning algorithm on the iOS platform that is able to infer the location of a tap on the surface. The vibration generated from the tap reaches the device and is read using the accelerometer, gyroscope and microphone. Taps from different locations generate different impulse responses due to variable distance travelled, reflections, scattering and dispersion in the medium, and boundary conditions of the surface. The impulse response captured by these acoustic and piezoelectric sensors is studied to extract feature sets that best represent the wave propagation, and differentiates signals from different locations, allowing for tap localization without the knowledge of surface parameters. The goal is to achieve a portable solution with a small training phase, minimal detection time, and one that is robust to the surrounding background acoustics, vibration and random noise.

Figure 1.1: Inference of user-tapped location on the surface

In this thesis, we discuss prior research and the theory behind sound propagation and localization in Chapter 2. Chapter 3 discusses the problem of tap detection using sensors available on iOS platform. A thorough study of various sensor detection corresponding to a tap event and the rational behind feature selection is also presented. We also present the tap classification algorithm central to this thesis and the results in Chapter 3. Chapter 4 discusses a sensor fusion algorithm to make the tap detection adapt to background noise. Finally, we discuss the problem of contiguous tap detection and provide a preliminary analysis and results in Chapter 4.
Chapter 2

Background

2.1 Prior Research

There exists several signal analysis and machine learning techniques that solve related tap localization problems [2, 3, 4, 5]. Some approaches apply multiple sensors, and some use more sensitive piezoelectric sensors application, which provide relevant background for our research. [2] applied cross-correlation combined with a weighted averaging of training locations, and a thresholding method, to interpolate discrete tap locations using a Knowles accelerometer. Their algorithm yielded good results, but was sensitive to spatial distribution of training locations. The threshold approach also required a predetermined value, which may vary between surfaces and tap types, leading to a lower detection rate. They also attempted to use a forward and backward propagation neural network with radial basis function to predict the tap location given the acoustic signal [2]. However, they found the calibration to not be reliable if the device placement changed relative to the boundaries of the surface, leading to either lengthy calibration phase or poor detection, as well as over-fitting of the data. As a second approach, they tried a reverse algorithm where they trained the network to generate the corresponding waveform given a (x, y) coordinate, which would essentially enable them to generate pseudo training data that was consistent. They did not complete their neural network investigation due to lack of expertise in the field and time constraints.

Nisha [3] explored few techniques such as time delay of arrival(TDOA) and cross-correlation to implement an acoustic tap tracker with multiple sensors (Polyvinylidene fluoride (PVDF) sensors) combined with signal amplification. The algorithm applied a weighting and thresholding method on the differences in cross-correlation peaks between the sensors, calibrated using template signals. The implementation yielded good result for low frequency knuckle tap, but performed poorly on hard taps, and taps located closer
than 10 cm apart [3]. [6] applied a similar time differential approximation and spectral analysis to detect knuckle, fist bang and metal tap contact on glass surface by mounting four sensors on four corners behind the tappable surface. The performance was degraded by strong dispersion affect in glass, and non-tap generated acoustic background sound, however yielded good enough results to build store front graphical display interface using knock tap detection [6].

A project called TAI-CHI [4] explored various tangible acoustic interface solution with application to Human Computer interaction such as electro-acoustic musical instruments, large-scale displays and so on [2]. Majority of the approaches investigated as part of TAI-CHI used time reversal, acoustic holography, combination of audio and visual detection and tracking, and time delay of arrival techniques with multiple sensors [7, 5], which can not be used with a single sensor approach. However, it laid down important foundation to future research opportunities to develop novel touch based user applications. In general, acoustic localization has been extensively studied in literature to solve for multiple sensors sound-localization, utilizing time-reversal and time delay of arrival technique, often augmented by various filtering, probabilistic model, and speech enhancement algorithms [8, 9, 10, 7]. However, single sensor-solution to acoustic localization is more desirable for developing software solution for most portable electronic devices due to limited power, memory or sensors availability. In this thesis, we present such a tap detection solution based on iOS platform that is able to infer tap locations without any prior knowledge of sensor location or surface parameters.

There has also been several security related research that studied inference of keystrokes from a nearby keyboard using accelerometer and acoustic sensors [11, 12, 13]. This was achieved by using accelerometer recordings corresponding to key press events from an Apple wireless keyboard placed 2 cm away from the phone in [12]. They trained a Neural Network using features such as mean, max, min, FFT, MFCC from accelerometer. However, due to the limited operational frequency of iPhone accelerometer, which is 100Hz, the neural net only achieved a detection rate of 25.89%. Their second approach used the same feature vector along with relative position of subsequent pairwise key-presses, trained with 150 key strokes per letter at random order and timing - constituting of 3900 distinct events, combined with word matching. This resulted in slightly better performance but varied between 91% to 65% detection rate depending on the left/right or far/near positions and the size of the data set. The same problem was solved in [14] by using acoustic data to train a neural network, where a PC microphone was used to listen to key-stroke events sampled at 44.1kHz. The FFT of the acoustic sample was used together with pairwise key pressing event to train the neural net with 100 key press sam-
Chapter 2. Background

7

samples per key. They managed to achieve a detection rate of 79% with the acoustic feature vector with the same amount of training. [11] improved upon the performance achieved by [14] by using cepstrum instead of FFT as feature vector together with unsupervised learning using Mixture of Gaussian estimation. The algorithm also incorporated a probabilistic model that biased the prediction based on the prior key detection event using Hidden Markov model, augmented by a language model to further improve the accuracy. These investigations while being reasonably successful, require a large training phase - training of a neural net with precise knowledge of relative key positioning of various key-boards, or clustering simulation using mixture of gaussian. Moreover, the performance degrades with distance and duration of detection. A solution that requires significant training phase can not be applied to the problem of detecting taps on a surface within the context of a portable user-application, as the algorithm needs to calibrate quickly to the new surface so that the client application can start using the detection software.

2.2 Acoustic Wave Propagation And Analysis Techniques

The surface waves generated due to a tap event can be modeled as Rayleigh waves [15, 16] which consist of both transverse and longitudinal waves, mostly confined to the surface. The energy of Rayleigh wave propagates radially from the source. The equation of wave propagation is given by [17, 16]:

$$\delta z(r,t) = A_t \sin(kr \pm \omega t)$$
$$\delta r(r,t) = A_l \sin(kr \pm \omega t)$$
$$A_{rayleigh} \propto \frac{1}{\sqrt{r}}$$ (2.1)

Surface waves can also contain Love Waves - which rise due to horizontal shear in the elastic material and also propagates radially. The solution to equation of motion in an elastic material of thickness $H$ is given by:

$$\phi(r,t) = A * \exp(-kd_{\text{shear}} \sqrt{1 - \frac{c^2}{\beta^2}}) \sin(kr \pm \omega t)$$ (2.2)

where the amplitude also falls off as $\frac{1}{\sqrt{r}}$ [17, 16]. Depending on the thickness of the medium, elasticity and reflection coefficient at the boundaries, various degrees of reflec-
tion, scattering and dispersion rises leading to different modes of vibration with variable frequency dominance. Therefore, the wave characteristics generated at a given location $(x, y)$ varies from surface to surface, and with boundary conditions. The goal of extended-touch is to use machine learning techniques instead of solving for wave propagation and hence develop a solution that will be adaptable to any surface dimensions or characteristics.

There are various well-known techniques that are used throughout literature to study acoustic signal localization: Image method [18], Time Delay of Arrival (TDOA) [2, 19], Time Reversal [19, 2, 20, 7, 10], and Cross-correlation algorithm [3, 2]. In this section, some of these methods that provided relevant foundation research are briefly discussed along with their applicability to solve the problem of extended touch.

### 2.2.1 Time delay of arrival

Time Delay of Arrival (TDOA) uses multiple spatially separated sensors, and accurate time delay estimation between them to solve for the source [2, 7], as illustrated in Figure 2.1.

\[
\sqrt{(\Delta x_1)^2 + (\Delta y_1)^2} - \sqrt{(\Delta x_2)^2 + (\Delta y_2)^2} = v\Delta t_{12}
\]
\[
\sqrt{(\Delta x_1)^2 + (\Delta y_1)^2} - \sqrt{(\Delta x_3)^2 + (\Delta y_3)^2} = v\Delta t_{13}
\]

where $\Delta x_i = x - x_i$, and $\Delta y_i = y - y_i$ (2.3)

However, if $v$ is unknown, more than three sensors are required. Moreover, the success of this method is highly dependable on the accuracy of the time delay estimation, and positioning information of the sensor. For this project, the exact position of various sensors on iOS devices vary from device to device, and are unknown. The time delay

![Figure 2.1: Time delay of arrival from source $(x, y)$ to the sensors](image-url)
between sensors can be estimated using peak-to-peak cross-correlation analysis on the detected signals. There are multiple microphones on iOS devices, however the APIs only provide a combined acoustic signal. The impulse response corresponding to a tap event detected by gyroscope, accelerometer, and microphone is shown in figures 2.2, 2.3, and 2.4.

Figure 2.2: Gyroscope detection for knock tap at (-10,0) on glass
Figure 2.3: Accelerometer detection for knock tap at (-10,0) on glass
The figures demonstrate that gyroscope and accelerometer x-y axes does not show significant detection past noise for a tap event. Therefore, only accelerometer z-axis and microphone detection can be utilized to develop a tap detection algorithm, which is an insufficient number of sensors to perform TDOA estimate. Moreover, TDOA performs poorly in the presence of reflection, which can easily rise from the surface waves reflecting off the boundaries of the device or other obstacles on the surface, and dispersion which leads to very complex wave-propagation characteristics in solids due to different modes of vibration and corresponding velocities [7]. Finally, sensitivity and resolution of these two sensors significantly differ, along with the type of waves detected by the two sensors (surface waves vs acoustic waves), thus leading to different speed of propagation, hence making it more difficult to perform TDOA estimation to solve for tap localization.

2.2.2 The image model

The Image method is a well-known technique where the location of the tap is solved by modeling the reverberation period of the acoustic signal [18, 21]. In a rectangular cavity, given a point source of vibration at location $X$ and a microphone at $X'$, the boundary
conditions of wave propagation can be satisfied by creating a symmetric image of the source. For a non-rigid wall, the impulse response of the tap can be modeled as: [18]

\[ p(t, X, X') = \sum_{p=1}^{8} \sum_{r=-\infty}^{\infty} \beta^6 \times \frac{\delta(t-(|R_p+R_r|/c))}{4\pi|R_p+R_r|} \]  

(2.4)

where \( R_p \) represents the distance between microphone to source and its symmetric image locations on \( x - y \) plane as shown in Figure 1 in [18]:

\[ R_p = (x \pm x', y \pm y', z \pm z') \]
\[ R_r = 2(nL_x, mL_y, nL_z) \]  

(2.5)

where \((L_x, L_y, L_z)\) are the dimensions in \( x, y, z \) directions, and \( \beta \) is the refractive index at the six walls, with the idealized assumption that the coefficient \( \beta \) is same for all the boundaries. The reverberation period can then be calculated from the impulse response using integrated tone-burst method:

\[ E(t) = k \int_{t}^{\infty} p^2(\tau) d\tau \]  

(2.6)

where \( p(\tau) \) is the impulse response from Equation 2.4.

The image method requires a prior knowledge of surface dimensions and coefficient of reflection and absorption, hence can not be used to develop a unified algorithm that works for an arbitrary surface, for which prior knowledge of surface parameters is not known. Moreover, most probable surfaces for our application will be rectangular half surface with various elastic properties leading to complex boundary conditions and wave dynamics. Hence most of the simplified assumption made by the image model no longer holds. Analysis of sensor data, example impulse response to tap events detected by microphone on iPhone is presented in figures 2.4 and 3.1, show that the acoustic signal from taps is complex and does not have a clear reverberation period due to various degree of reflection and refraction at the boundaries. Moreover, it is also possible that in practical application, the surface such as a table will have more objects placed on it which will also lead to more complex boundary conditions for wave-propagation. Therefore, the image model may not provide an effective way to solve for sound localization of user-tapped locations using sensor on smartphones that could quickly adapt to any surface for such application.
2.2.3 Cross-Correlation

Given two signals \( s_1(t) \) and \( s_2(t) \), the maximum cross-correlation \( X \) between them is given by:

\[
X(s_1(t), s_2(t)) = \max_{\phi} \frac{\phi_{s_1s_2}(t)}{\sqrt{\phi_{s_1s_1}(0)\phi_{s_2s_2}(0)}}
\]

\[
\phi_{s_1s_2}(t) = \int_{-\infty}^{\infty} s'_1(\tau - t)s'_2(\tau) d\tau
\]

\[
s'(t) = s(t) - \overline{s}
\]

The absolute value of \( X \) lies between \([0, 1]\), 0 for not-correlated and 1 for highly correlated or the same signal. Therefore, the maximum cross-correlation value between two signals can be used as a measure of similarity between two impulse response samples. As discussed earlier, taps at different locations generate unique impulse response corresponding to the tap locations due to various degree of surface-wave interaction and propagation boundary conditions. The algorithm presented in this thesis uses this property and employs the maximum cross-correlation between two signals as a measure of distance, and utilizes this knowledge to tell apart taps at different locations.

2.3 Machine Learning Techniques

Solving for tap localization using wave-propagation analysis can become significantly difficult and not-portable due to all the surface parameters that must be accounted for which will change depending on the application and usage of extended touch. Therefore, as part of the thesis several machine learning techniques were investigated that could use sensors data as the feature vectors to infer a tap location, which formed the basis of extended touch detection.

2.3.1 K Nearest Neighbor

K Nearest Neighbor (KNN) is a supervised machine learning technique used mainly in clustering problems. KNN is performed by first dividing the sample space into several known classes, each of which are represented by a set of features. The classification of a test sample is performed by analyzing the neighbors in the feature space. For extended-touch, each tap locations to be detected constitutes a discrete output class. A new tap event can then be classified into one of the output locations by finding the best-match
cluster and provide the cluster-identifier as the output location. The success of nearest-neighbor classification is dependent on good feature selection that can maximize distance between two distinct classes and minimize spread within the same class. Moreover, the performance can also vary depending on the number of neighbors (k) considered, pre-processing of data, presence of noise and data sparsity.

2.3.2 Regression Analysis

Regression analysis is technique that can be used in classification or function prediction problem. There are two types of regression analysis that were used as part of extended-touch investigation - linear and logistic regression. Logistic regression can be used to solve for binary classification problem which can indicate how well the picked feature is able to differentiate two classes - i.e. tap locations. A linear regression analysis technique is employed later to predict a function that can detect a tap between two different known tap locations, which will be further discussed in the next chapter.
Chapter 3

Tap Detection and Classification

Given an understanding of wave propagation in surfaces generated due to a tap event, combined with the knowledge of existing sound localization techniques and prior research on related tap or vibration detection, we set out to solve the problem of extended touch detection using sensors available on iOS platform. We do this by first taking a close look at each of the sensor data, and subsequently developing a method to use the detection to help predict where in the surface the tap has occurred.

3.1 Sensors and Tap Detection

For a particular tap location, data from different sensors are independent. If multiple sensors could be utilized to detect a tap, their combined detection could lead to better classification confidence and help resolve confusion between classes. The first objective is to pick at least one feature that minimizes variance within a given class (tap location), and maximizes distance between different classes. The ideal sensor would be one that captures a clean impulse response corresponding to various user tapped locations. The different motion and acoustic sensors available on the iOS platform and their detection of taps on the surface is studied in this section.

3.1.1 Microphone

The microphones on iOS devices vary in model, specifications and in quantity between generations and particular models. On the iPhone 4 and iPad, there are two microphones - and are known to have a sharp cutoff in frequency response at 20kHz [22]. On iPhone 5, there are three microphones that support HD Voice - one in front and back near the camera, and one at the bottom - have double the frequency and spectrum width as of the
previous models [23]. The extra microphones are used for noise cancellation, and are not accessible as separate audio units. High quality low latency audio input can be obtained by sampling at the recommended rate of 44.1kHz.

Microphone detection of tap events at different location for different types of example taps is shown in figure 3.1. The signals captured by the microphone shows clear impulse response corresponding to each tap.

Next, a cross-correlation analysis was performed to study whether microphone detection can distinguish between taps at different locations. Figure 3.2 shows sample plots of maximum cross-correlation of two taps at (10,0) cm and (20,0) cm with template taps at various locations. The figures illustrate that maximum cross-correlation of microphone signal between taps at the same location is much higher relative to taps that differ in location or type (i.e. knock taps compared to soft taps). Therefore, microphone has high enough resolution to capture various surface wave propagation interaction leading to good cross-correlation values that can be used to differentiate taps at different tap locations. Moreover, in Chapter 4, we will show that the cross-correlation of audio signal can reliably indicate similarity between taps at the same location even in the presence of background acoustic and various levels of gaussian noise.
Figure 3.1: Microphone detection of tap impulse response for various tap types
Figure 3.2: Max Cross-Correlation of sample taps with different template taps. Microphone detection shows taps at the same location are highly correlated.
3.1.2 Accelerometer

Accelerometers on iOS devices and other consumer products have been used for a wide variety of applications - user motion and activity detection, shake, tap, vibration or sudden impulse detection, aiding GPS in navigation and so on [24]. Various quantities such as mean, max, and min amplitude, mean wave period, interval average acceleration as function of time, interval root mean square acceleration as function of time, trim-mean acceleration vs time, power spectrum density vs frequency, and various signal analysis techniques could be used to extract information from an accelerometer signal [25, 26]. However, one of the constraining factor of accelerometers is their max-allowable frequency and dynamic range available on a particular platform. Apple uses STMicroelectronics LIS302DL piccole accelerometer, which measures acceleration in units of g, and gives a maximum frequency of 100 Hz, resolution of 0.018g and dynamic range of +/-2.3g [27]. The x and y axes of accelerometer detected weak pulses for some taps - example shown in Figure 3.3, due to presence of shear component of wave generated from the tap.

![Accelerometer x-axis detection](image1)

(a) accelerometer x-axis detection

![Accelerometer y-axis detection](image2)

(b) accelerometer y-axis detection

Figure 3.3: Accelerometer detection in x and y axes for soft tap (10,0)

while not detecting anything but noise for others as illustrated in Figure 3.4. Accelerometer z axis data shows a clear impulse response detection shown in Fig-
Chapter 3. Tap Detection and Classification

20

(a) accelerometer x-axis detection

(b) accelerometer y-axis detection

Figure 3.4: Accelerometer detection in x and y axes for knock tap at (-10,0)

(a) accelerometer z-axis detection

(b) accelerometer z-axis, soft tap (10,0)

Figure 3.5: Accelerometer z-axis detection
Chapter 3. Tap Detection and Classification

Figure 3.5, due to a stronger transverse component of the surface wave. However, the accelerometer detection of soft tap from Figure 3.5-(b) is not as prominent with good resolution as the impulse response recorded for the knock tap by the accelerometer z-axis. Although accelerometer z-axis shows a cleaner detection for at least a knock tap, the 100Hz resolution is still too low to capture all the wave propagation characteristic into the impulse response, which becomes clear once we perform cross-correlation analysis on the accelerometer z-axis detection of various tap samples.

Figure 3.6 shows the maximum cross-correlation values of accelerometer z-axis recording of a test tap at location (20,0) with different accelerometer z-axis recording of taps at different tap locations, which also includes a different tap sample at the same location (20,0). It can be seen from the figure that the maximum of the maximum-cross-correlation values occur at 0.618 against another tap at location (-10,0). However the test tap was location at (20,0) which is not the same as the best matched tap location (-10,0) using maximum cross-correlation value, while the maximum cross-correlation with another tap at the same location (20,0) was not the highest.

Figure 3.6: maximum cross-correlation of knock tap at (20,0) with template taps at various locations

Therefore, cross-correlation analysis of accelerometer signal does not show a strong correlation between signals from the same tap location, and is confused with taps at other locations. Hence, even though the accelerometer detects a distinct impulse response cor-
responding to a tap event, the sensor resolution is not high enough to capture informative wave-propagation characteristics to be useful in a cross-correlation analysis, thus can not be used as a measure of distance between tap signatures at different locations. The accelerometer is also very sensitive to ambient vibrations and noise, thus must also be carefully considered with filtering, in order to be utilized correctly to infer existence of a tap event.

### 3.1.3 Gyroscope

Gyroscope is a MEMS sensor that measures change in orientation. The gyroscope APIs on iOS platform provide rate of rotation about x, y, and z axis of the device in radians per second. The surface vibration from a tap event could result into angular vibration leading to gyroscope detection. Therefore, the gyro-detection of a tap event was also studied as part of initial investigation. Figure 3.7 shows gyroscope detection for tap at (-10,0). The gyroscope data was found to be noisy and barely showed any detection, which was expected as vibration from a tap is not strong enough on glass to cause angular displacement of the device.

The investigation revealed that microphone and accelerometer show clear impulse response for a tap event, whereas gyroscope does not provide any significant detection. Moreover, microphone detection of a tap event was found to be of high enough resolution to capture a good impulse response which showed a high correlation across signals generated by tapping at the same location relative to cross-correlation between taps from different location. The accelerometer z-axis shows a impulse response detection corresponding to a tap, however can not be used for cross-correlation analysis to differentiate taps from different location due to the low resolution of 100Hz available on iOS devices. Therefore, the final sensor selected to be used for tap classification algorithm is microphone recording of a tap event. However, even though accelerometer z-axis signal can not be used for tap classification, it can still be used to infer when a tap has occurred. In the next section, the extended-touch classifier is discussed which will use cross-correlation of microphone signal as the feature vector to detect and classify taps on various surfaces.
Figure 3.7: Gyroscope detection for knock tap at (-10,0) on glass

(a) gyroscope x-axis

(b) gyroscope y-axis

(c) gyroscope z-axis
3.2 Tap Classification - Binary Classifier

Following an extensive analysis of sensor recording of taps on glass surface and feature selection consisting of accelerometer and microphone detection of taps, next step was to develop an algorithm that can use these features to classify discrete taps on any surface. In order to achieve this, we first focused on developing a binary classifier that can tell apart two different taps based on maximum cross-correlation match using microphone detection of a tap event.

As discussed in Chapter 2, the maximum cross-correlation $X$ between two signals $s_1(t)$ and $s_2(t)$ given in equation 2.7 can be used as a measure of similarity between the two signals. Moreover, in the last section, it was shown that maximum-cross-correlation value between microphone detection of taps show high correlation for taps at the same location, and low correlation for taps at different locations. This relation forms the basis of the algorithms developed to detect taps on a surface.

3.2.1 Nearest Neighbor Classifier

We developed a nearest neighbor classifier that used cross-correlation match as the distance measure to detect and tell apart taps generated at two different locations. A simple iPhone application was developed to implement the binary classifier using the maximum cross-correlation to infer tap locations. To detect taps from any two locations, the binary classifier algorithm works as follows:

- **Training phase:**
  - Pick two tap locations for the binary classification test setup.
  - Store two template waveforms corresponding to microphone recordings of taps at each of the two tap locations.

- **Detection:**
  - Record accelerometer and microphone signals in real-time.
  - Analyze accelerometer data continuously. If the accelerometer amplitude is past a pre-set threshold, a tap has been detected.
  - Once accelerometer detects a tap, use microphone signal recorded for that tap window as the sample tap to be classified.
  - Compute maximum cross-correlation $X$ of the sample tap signal with the training tap microphone signals.
Chapter 3. Tap Detection and Classification

- Pick the training tap location that gives the largest maximum cross-correlation value \( X \) as the best match, if it has passed a pre-defined threshold check.
- Output the tap location of the best matched template as the predicted location for the detected test tap.

![Glass table](image1.png) ![Metal sheet](image2.png) ![Hollow wood](image3.png)

(a) Glass table (b) Metal sheet (c) Hollow wood

Figure 3.8: Different surfaces used to test classification on iPhone

The application was tested on three different surfaces - glass rectangular table, metal sheet on a flat surface, and a square wooden table, as shown in Figure 3.8.

On each of the surfaces, five different pair-wise configurations of tap locations were tested using the binary-classifier algorithm discussed above. The setup of the configurations is shown in Figure 3.9, where the red filled circles are the tap locations tested by the binary classifier for each of the setup.

Results of the live classification on the iPhone of 100 test knock taps on the glass table for the 8 tap locations is shown in Table 3.1, which had an average success-rate of
Chapter 3. Tap Detection and Classification

Figure 3.9: Tap location pairs for live classification test on iPhone

92%. The configuration with the highest error was the symmetric horizontal setup shown in Figure 3.9f.
Table 3.1: Glass table

<table>
<thead>
<tr>
<th>Tapping pair</th>
<th>Correct</th>
<th>Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric (A)</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Asymmetric (B)</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Symmetric (horizontal)</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Symmetric (diagonal A)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Symmetric (diagonal B)</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>92</td>
<td>8</td>
</tr>
</tbody>
</table>

The results of 100 knock taps tested on the hollow wooden table and the metal sheet is shown in tables 3.2 and 3.3 respectively. The performance of the binary classifier was the worst on the hollow wooden table - 89%, where the symmetric configurations had higher confusion rate. The performance on the metal sheet was highest out of all the surfaces - 97%.

Table 3.2: Hollow wood table

<table>
<thead>
<tr>
<th>Tapping pair</th>
<th>Correct</th>
<th>Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric (A)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Asymmetric (B)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Symmetric (horizontal)</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Symmetric (diagonal A)</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>Symmetric (diagonal B)</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>89</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 3.3: Metal sheet

<table>
<thead>
<tr>
<th>Tapping pair</th>
<th>Correct</th>
<th>Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asymmetric (A)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Asymmetric (B)</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Symmetric (horizontal)</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Symmetric (diagonal A)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Symmetric (diagonal B)</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>97</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.4: Aggregated Results from three surfaces - glass, wood, metal, 8 configurations of pairwise taps

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>278</td>
<td>22</td>
</tr>
<tr>
<td>Success Rate</td>
<td>92.66%</td>
<td></td>
</tr>
</tbody>
</table>

The average detection rate of the binary classifier for the 8 pair-wise tap locations was 92% across all three surfaces using the cross-correlation match. Therefore, the performance of the tap detection algorithm for a set of two tap locations can vary depending on the surface types, the amount of vibration caused by the taps, and finally the consistency between the taps themselves. However, the algorithm performs reasonably well across various surfaces with a very small training step and detection time with minimal signal processing, thus ideal for portable smartphones where the processing power and memory allocation reserved for each application can be quite constrained by the operating system.
3.2.2 Logistic Regression

To establish a benchmark, a logistic regression classifier was implemented to compare performance with the nearest neighbor binary classifier. The feature vector used for logistic regression was also the max cross-correlation between two signals. Therefore the algorithm can be described as follows:

- Store two template signals representing the two locations to be classified.
- Collect $N$ number of trainings samples, and compute their cross-correlation which is to be used as a feature vector.
- Train a binary logistic classifier using gradient decent and L2 regularizer.
- Test the classifier on a new validation set containing $M$ number of samples for the two locations, using cross-correlation as the feature vector.

The equations for the likelihood and gradient decent for logistic regression is shown below:[28]

$$\begin{align*}
\mathbf{x} &= \text{input vector which contains maximum cross-correlation feature} \\
Z &= \mathbf{w}^T \mathbf{x} + w_0, \text{ and the likelihood function } l(\mathbf{w}) \\
l(\mathbf{w}) &= \sum_n t^n \log p(C_1|\mathbf{x}_n, \mathbf{w}) + (1 - t^n) \log p(C_0|\mathbf{x}_n, \mathbf{w}) \\
p(C_1|\mathbf{x}, \mathbf{w}) &= \frac{\exp(Z)}{1 + \exp(Z)} \\
p(C_0|\mathbf{x}, \mathbf{w}) &= \frac{1}{1 + \exp(Z)} \\
w_i^{n+1} &\leftarrow w_i^n + \alpha [-\lambda w_i^n + \frac{\partial l(\mathbf{w})}{\partial w_i}] \quad (3.1)
\end{align*}$$

The error rates for both training and test set from the simulation is shown in Figure 3.10. The regularizer and learning rate was varied to study the performance of the learning method as function of the parameters. The success rate of detection after many iterations was 100% on both the training and validation set. Although, the data set was small, equal number of samples were used for both training and testing phases. The success-rate of logistic regressor demonstrates that maximum cross-correlation is a good feature that can linearly separate two classes. This binary regression classifier can be extended to a K-class logistic regressor but with significant amount of added computational complexity and processing time. To extend the binary logistic regressor
to K-class classifier, one would have to train k-functions corresponding to each of the $C_k$ classes with

$$z_k = w_k^T x + w_{k0}$$

where one would have to optimize for k different w vectors. Moreover, training a k-class logistic regressor will require more training samples to avoid over-fitting the data, and introduces additional training time due to multiple iterations. Therefore, even though the detection rate with logistic regression is high, for the problem of detecting tap on a smartphone, it is more desirable to implement a method that requires least amount of processing time. Training phase for the nearest neighbor algorithm is small, while the success-rate is reasonably high making it a better solution for an online classification that has to quickly train and detect taps using a smartphone.
3.3 Multi-class Classification, Training and Test Results

Following the high success rate with the binary classifier with nearest neighbor method, next step was to detect discrete taps from any number of tap locations. In this section, we propose a multi-class classifier, where each class is a distinct tap location that is to be detected by the algorithm, and present the results when applied to detect 17-different tap locations. The proposed multi-class classifier is a K Nearest Neighbor (kNN) algorithm, an extension of the binary nearest neighbor classifier. KNN is a supervised learning algorithm in machine learning where a test sample is classified into one of the training classes by finding and ranking it’s nearest neighbors. The algorithm uses maximum cross-correlation between test location signal \( v_{ts} \), and the template signals in the training database \( \{(v_{tr_i}^l, t_{tr_i})\} \) as a measure of distance, to choose the nearest neighbor.

\[
\begin{align*}
  \text{let } T &= \{ \text{set of training data set} \} \\
  v_{tr_i}^l &= \text{microphone impulse response for tap } t_{tr_i}^l \\
  t_i &= \text{tap location label}, (v_{tr_i}^l, t_{tr_i}^l) \in T \\
  v_{ts} &= \text{microphone data from test location} \\
  X(v_{ts}, v_{tr_i}^l) &= \max ((v_{ts} \ast v_{tr_i}^l)(t)) \text{ cross-correlation} \\
  L &= \{v_{ts}^j\}, \text{test locations} \quad (3.2)
\end{align*}
\]

The predicted location \( t_{ts} \) for \( v_{ts} \) is determined by the maximum votes from it’s k-nearest neighbors. This approach makes the algorithm more robust to noise, mis-labelled data, and outliers in the training set. KNN yields good results if the data in the training set is uniformly distributed. We ensure this by picking the same number of samples from each training tap locations. Therefore, our algorithm can be described as follows:

- \( \forall \) test sample \( v_{ts} \in L \), compute \( X_i(v_{ts}, v_{tr_i}^l), \forall v_i \in T \)
- sort \( X_i \) in descending order
- count up the votes from each \( v_i \in T \)
- pick \( V_k = \{t_{tr_i}^l\} \) corresponding to k largest \( X_i \)
- pick the \( t_{tr_i}^l \) with the largest vote. If two classes have the same vote, then pick the one with largest \( X_i \)
The kNN algorithm was implemented in Matlab, and tested offline using iPhone microphone data collected from 17 different tap locations. The simulation first picks at least one template signal for each output class (a discrete tap location) that is to be recognized by the classifier from the data set. The remaining taps recordings then forms the test set to be classified using the kNN algorithm. The error rate was calculated by comparing predicted location of the test tap samples outputted by the classifier, with the true tagged locations. The kNN was tested on data from knuckle taps and soft taps separately, and results are summarized in table 3.5. The results show that kNN has a high success rate for knuckle taps, while yielding a low detection rate for soft taps. Table 3.6 presents the break down of misclassified soft taps for different distance errors between the true and predicted locations. The results demonstrate that majority of the misclassified soft taps are confused with taps that are within about 5 cm radius.

A second set of data set was tested where 2 soft tap classes and 15 knock tap classes were combined with 1 template impulse response per class. The results are summarized in table 3.7, where the combined detection rate was 90% and all the soft taps were correctly detected. Table 3.8 shows the results using training set where only one training sample per location was used without differentiating between the tap types - soft taps or hard taps. The failure rate of 25% occurred because taps from the same location are highly correlated given the taps are of the same type - i.e. a knock tap is not highly correlated with a soft tap from the same location, but shows a strong correlation with another knock tap at the same location.

### Table 3.5: Classification of 17 locations, 1 template/location

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Knuckle Taps</th>
<th>Soft Taps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Output Locations</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>Number of Test Samples</td>
<td>86</td>
<td>49</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>92%</td>
<td>63%</td>
</tr>
<tr>
<td>Misclassification Error</td>
<td>8%</td>
<td>37%</td>
</tr>
</tbody>
</table>

### Table 3.6: Misclassification of Soft Taps

<table>
<thead>
<tr>
<th>Distance between true and predicted location [cm]</th>
<th>Number of test samples misclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 3.7: Mixture of Taps, 1 training/(x,y)/type

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Soft &amp; Knuckle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Output Locations</td>
<td>17</td>
</tr>
<tr>
<td>Number of Test Samples</td>
<td>61</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>90%</td>
</tr>
<tr>
<td>Misclassification Error</td>
<td>10%</td>
</tr>
</tbody>
</table>

Table 3.8: Mixture of Taps, 1 training/(x,y)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Soft &amp; Knuckle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Output Locations</td>
<td>16</td>
</tr>
<tr>
<td>Number of Test Samples</td>
<td>62</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>75%</td>
</tr>
<tr>
<td>Misclassification Error</td>
<td>25%</td>
</tr>
</tbody>
</table>

Therefore, we expect the kNN algorithm to detect both knock and soft taps with high accuracy as long as the training database includes both soft and knock tap impulse response templates. However, depending on the spatial resolution of the soft taps we wish to detect, the success rate would be low. We expect that the results of kNN classifier may improve with more training templates per class, such that the combined prediction would be more robust.
3.4 Methods for combining Multiple Templates and Results

While the kNN classifier gave a good results for multi-class classifier simulated in Matlab, run on tap data collected from 17-distinct tap locations on the iPhone, the implementation of the classifier on smartphone led to degraded performance due to inconsistency in the user taps. The data for the simulation was collected in a very controlled setting, where the tap location for each tap-class was marked down, and test samples were collected by precisely hitting at the exact original marked locations. However, in practice, a user is unlikely to mark down the exact location, and therefore their taps at a location will vary from tap to tap. Moreover, the intensity of the tap may also vary depending on whether the user is consistent while using the tap-prediction solution. In this section, various methods to combine multiple templates for a given tap class is investigated with the expectation that the performance of real-time tap classification may significantly improve. Moreover, analysis of detection time and success rate is presented for each approach.

3.4.1 Inconsistent tap

To measure the impact of inconsistent taps, a new experiment was carried out where participants were asked to casually tap 50 times at 2 locations. The process was repeated 31 times, resulting in 3100 individual taps. In each one of the 31 repetitions, the iOS device was placed at a randomly selected location on the table, and the table itself was randomly chosen among three different wooden tables and a glass table.

From this experiment, it was found that the accuracy dropped from 92% to 82% when only one template is used for each output tap location. However, the success rate of the classifier increased with the number of templates stored in each location. This performance improvement can most likely be attributed to the rational that having multiple templates per output class allows the algorithm to capture the variability and differences between user taps for the same tap location that may differ ever so slightly.

Let $M$ be the number of templates recorded for each of the output tap location $t_i$ during the training phase, and the $j^{th}$ template for location $t_i$ is denoted as $v_{ij}^{tr}$. Let $v^{ts}$ denote the microphone waveform that needs to be classified. The average cross-correlation of the test tap $v^{ts}$ relative to template tap location $v^{tr}_i$ is denoted by $X_i$, 

...
which is calculated relative to each training tap locations $t_i$ as follows:

$$X_i = \frac{1}{M} \sum_{j=1}^{M} \max\left( \sum_{m=-\infty}^{\infty} v_{ts}[m]v_{t_j}[m+n] \right)$$  \hspace{1cm} (3.3)

After $X_i$ is calculated relative to each tap output class $t_i$, the predicted tap location for the test tap $v_{ts}$ is determined as follows:

$$t_{ts} = \arg \max_i X_i$$  \hspace{1cm} (3.4)

Figure 3.11 shows the classification success rate versus the number of templates used for each of the output tap classes (i.e. each tap location to be predicted) during the training phase by the algorithm. From this data-set, it was found that 5 templates for each tap location during the training phase results into a good enough accuracy while still keeping the training phase not too tedious for the user.

![Figure 3.11: Classification Rate vs. Number of Templates per Location](image)

However, as the number of templates increase for each location, the detection time also increases linearly due to the number of cross-correlation analysis required to output a predicted location for the sample tap. In a real-time scenario while computation is
being done on the iPhone, cross-correlation analysis with \(5 \times \#(\text{of output locations to be detected})\) can introduce significant performance lag, or cause memory issues, leading to a poor user experience. Therefore, as an alternate solution, we explore an algorithm that generates a representative waveform from multiple templates for each output tap class, thus requiring only one cross-correlation to be performed per class corresponding to the distinct tap location to be detected.

### 3.4.2 Speed Boost Using Centroid of Templates

To reduce the amount of computation needed with multiple-templates per tap location, one can pre-process the set of templates for each location, and generate a new representative template, which can then become the new training sample tap for that tap location class. One simple approach to generate the representative template signal is to time align the set of templates recorded for the same tap location in the training phase, and take the average of the set to be the final training tap template. Therefore, the final training tap sample for tap location \(i\) can then be calculated as:

\[
v_{tr}^i = \frac{1}{M} \sum_{j=1}^{M} v_{tr}^{ij}
\]

(3.5)

Using the representative template signal, the cross-correlation value of the test tap relative to the tap class is then given by:

\[
X^*_i = \max(\sum_{m=-\infty}^{\infty} v_{ts}[m]v_{tr}^i[m + n])
\]

(3.6)

Note that (3.6) will be equivalent to (3.3) if all templates at the location are perfectly aligned with each other. Figure 3.12 contains an example of two unaligned microphone waveforms for the same tap location. Next, there are multiple ways one can time align signals, out of which two methods are discussed below for the microphone data:

1. Onset-Based

2. Cross-Correlation

**Onset**

The onset method for waveform alignment depends on the observation that the onset of a tap has more energy than noise. Therefore, given two signals, one can deduce the
alignment point when the amplitude of the signal passes a pre-defined threshold limit. However, the onset-based method is not robust. In most practical situations, it is hard to pick a good onset threshold that robustly predicts the start of a signal. Therefore, we decided to use the second approach to time align templates.

Cross-Correlation

A more robust solution for time-alignment is to cross-correlate the calibration signals with respect to a base signal. The base signal is picked to be one of the tap signals from the set of training taps that are recorded for the same tap location. Therefore, given the time lag $l_{ij}$ for template $j$ at location $i$ (with respect to $t_{i1}$) is computed as:

$$l_{ij} = \arg \max_n (\sum_{m=-\infty}^{\infty} t_{i1}[m]t_{ij}[m + n])$$
After $l_{ij}$ is computed for all templates at location $i$, the aligned version of its $j^{th}$ template $t_{ij}^A$ is:

$$t_{ij}^A[n] = t_{ij}[n + l_{ij}]$$

$t_{ij}^A$ is then used in equation (3.5) to generate the centroid representing all templates for location $i$.

### 3.4.3 Performance Evaluation

In this section, the results from the three proposed methods for tap localization were measured for speed and accuracy, and compared against each other over 31 different data sets. The three methods are outlined below:

1. Method 1: Tier classifier with one template per tap location class
2. Method 2: Directly using 5 templates per location, computing cross-correlation relative to each template separately, and selecting the best correlation (3.4)
3. Method 3: Time align 5 templates to form a representative template for each tap location class, compute cross-correlation against the representative signal, and then obtain the best match

The results for the experiment are given in table 3.9. There would be a preprocessing time associated with finding the representative template (the centroid) for each location, whose runtime was not measured, but is expected to be small since this is done only once per tap class after the training phase.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
<th>Success Rate</th>
<th>Runtime (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One template per location. User only calibrates once</td>
<td>82%</td>
<td>7.30</td>
</tr>
<tr>
<td>2</td>
<td>Average of five cross-correlations per location</td>
<td>91%</td>
<td>3*39.41</td>
</tr>
<tr>
<td>3</td>
<td>Centroid of templates generated with xcorr alignment</td>
<td>89%</td>
<td>3*7.30</td>
</tr>
</tbody>
</table>

It can be easily concluded that using 5 templates becomes very expensive in terms of computing time and power, hence does not seem to be a good trade-off for smartphone
applications even though the success rate is higher. The centroid method is more reliable over method 1, with not much added computing overhead, hence seems to be a good trade-off. However, it is still tedious to ask the user to calibrate 5 times for each tap location, specially since they would have to repeat the process every time the phone is moved on the same surface or onto a different surface. Therefore, for the practical application of extended touch, it may still be desirable to keep the number of templates to 1 or 2 per training tap locations, even if the accuracy slightly reduces for the users who do not tap consistently, since it still performs quite well in most cases. We also observed that if the same user does the calibration training, and continues on to use the detection solution, the performance is better since the user tends to be more consistent. The accuracy of the tap detection does degrade if a different person performed the calibration phase. Moreover, as the user uses extended touch, he or she gets better at tapping more consistently, thus giving a better result over time.
Chapter 4

Sensor Fusion for a Robust Classifier

The k Nearest Neighbor algorithm to detect and localize taps using maximum cross correlation as the feature vector results in a relatively successful classifier. However, this task can become increasingly difficult in a noisy environment with various types of background sound. In this section, the effect of background sound on the performance of the classifier is studied, and a method is presented which is shown to improve the performance of the tap inference with background noise.

4.1 Tap detection in the presence of background sound

To study the effect of background sound on tap inference, we start by analyzing microphone and accelerometer data of knock taps while people are talking in the background. Two example data samples consisting of microphone and accelerometer signal corresponding to tap detection are shown in figures 4.1 and 4.2.

Each data sample consists of a time-series sensor recording of 2.5s length with continuous background sound. The actual tap occurs sometime within the time window. The accelerometer shows no detection until the actual tap occurs, while microphone’s recording of various background noise makes it difficult to narrow down the region where the actual tap has occurred. Therefore a peak in accelerometer reading could be used as a strong indication to detect a tap event and subsequently run the kNN classifier on the microphone signal.

As the first step of the analysis, we take a closer look at the tap classification error that would occur if accelerometer was not used to decide when to perform the cross-correlation to run kNN algorithm. To simulate tap detection algorithm running on a device every 10th of a second to output a detected tap and it’s location, a moving window of 0.1s was run over the test data set consisting of about 4 test samples per location, each
Chapter 4. Sensor Fusion for a Robust Classifier

Figure 4.1: Examples of tap detection with talking in the background at location (0,20)

containing 2.5s of sensor recording per sample, with talking in the background and a tap event sometime within the test sample, which gives 100 detection windows per tap location. The microphone signal from this detection window (treated as test samples in this case) was then used to compute maximum cross-correlation feature vector relative to each of the training taps, and classify using the kNN classifier with an added logic where no tap is detected if the goodness of fit (maximum cross-correlation) is below a pre-set threshold. The classification results is discussed for different example threshold values - 0.1 and 0.05, which corresponds to the maximum cross-correlation that must be achieved for a test sample to be considered a valid tap sample and classification.
The misclassification rate is then calculated as follows:

\[
\text{error rate} = \frac{n_{wTh}}{N_{totTh}} \quad \text{where} \\
\begin{align*}
n_{wTh} &= n_{\text{wrong}} | \text{goodness of fit} > \text{threshold} \\
N_{totTh} &= N_{\text{total}} | \text{goodness of fit} > \text{threshold}
\end{align*}
\] (4.1)

The above simulation was run over two data sets from two different experiments with the same setup but containing different tap locations. For both of these data sets, 2 training samples per tap class, and \( k = 2 \) was used for the kNN classifier. The training samples were clean tap signals with no background noise, where as the test samples contained talking in the background. The confusion matrix from dataset 1 computed over the sliding window consisting of essentially 325 test samples is shown in table 4.1
Table 4.1: Confusion matrix, talking dataset 1, threshold 0.1, 325 samples: 3 taps locations, 0.1s sliding window

<table>
<thead>
<tr>
<th>Predicted locations</th>
<th>True locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,20)</td>
<td>(0,20)</td>
</tr>
<tr>
<td>(10,0)</td>
<td>(10,0)</td>
</tr>
<tr>
<td>(-10,0)</td>
<td>(-10,0)</td>
</tr>
</tbody>
</table>

for threshold 0.1.

Figure 4.3: Error rate, threshold = 0.1, 0.1s moving window across all test samples, a total of 325 detection windows

The misclassification rate over the sliding window of detection algorithm is shown in figure 4.3 which shows that depending on whether the window coincided with a true tap event or not, the false detection rate could be high. The confusion matrix for the tap classes shows that there are substantial confusion between the tap classes in the presence of background sound even though a goodness of fit threshold is used.

A second experiment was run with talking in the background that included more tap locations, same test setup, and taps that are as close as 5 cm apart. The confusion matrix for data set with threshold 0.05 is shown in tables 4.2 and 4.3. The results from both experiments demonstrate that in the presence of background acoustics, even by applying a threshold approach, taps can be falsely detected even when there are no taps, and misclassified leading to a high confusion rate and classification error.

Now, we look at the slice of the microphone signal where the accelerometer does
Table 4.2: Confusion matrix, talking dataset 2, 7 tap locations, threshold 0.05, 0.1s sliding window

<table>
<thead>
<tr>
<th>True locations</th>
<th>(+10, 0)</th>
<th>(+10, +10)</th>
<th>(+15, 0)</th>
<th>(+20, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10, 0)</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>(10, 10)</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(15, 0)</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(20, 0)</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>(20, 20)</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(30, 30)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(5, 0)</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.3: Confusion matrix, talking dataset 2, 7 tap locations, threshold 0.05, 0.1s sliding window (continued)

<table>
<thead>
<tr>
<th>True locations</th>
<th>(+20, +20)</th>
<th>(+30, +30)</th>
<th>(+5, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10, 0)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(10, 10)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(15, 0)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(20, 0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(20, 20)</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(30, 30)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(5, 0)</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

show a peak and classify the test samples using the same algorithm. The algorithm now starts the microphone cross-correlation over a 0.1s window only once the accelerometer reading peaks 20% past the minimum of the maximum accelerometer reading amplitude measured during the training phase. The algorithm uses the same max cross-correlation feature vector and threshold value as before. The results of this algorithm given in table Table 4.4 show that the miss-classification rate drops to 0% for the same data set that was presented in table 4.1, and for the data set from experiment 2 whose confusion matrix was shown in table 4.2 and 4.3. The classification success rate for experiment 1 and 2

Table 4.4: Confusion matrix for dataset 1, tier classifier using sensor fusion, 3 tap locations, 0.1s detection window

<table>
<thead>
<tr>
<th>True locations</th>
<th>(0, 20)</th>
<th>(10, 0)</th>
<th>(-10, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 20)</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(10, 0)</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(-10, 0)</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
for all the tap locations was 100%.

Therefore, the proposed algorithm has two tier logic - detection and classification. First tier detects a tap event by measuring the z-axis accelerometer amplitude within a 0.1s moving window. If the accelerometer peaks above the pre-decided value, measured during training phase by taking the minimum of the accelerometer z-axis max amplitudes, the next 0.1s window of microphone reading is passed onto the 2nd tier which classifies the sample by employing the kNN classifier with a specified goodness of fit threshold. Application of this tier classifier logic that checks the state of accelerometer to decide whether to perform kNN classification gives a measurable performance improvement and reliable detection over the previous method that only relied on audio feature vector.

4.2 Misclassification rate with degrading SNR

In this section, a more in depth analysis is presented to gaze the performance gain obtained by using accelerometer to aid microphone detection of taps on the surface. To quantify the advantage, we study the rise in misclassification rate as the Signal to Noise ratio (SNR) degrades in the presence of background noise. The decrease in SNR is simulated by adding Gaussian white noise to the test samples by applying the Matlab function:

\[ \text{awgn}(<\text{test sample }>, <\text{desired snr }>, '\text{measured}') \]

The same datasets discussed in the previous section containing 2.5s sensor recording were injected with the gaussian noise, to obtain varying target SNR levels to simulate various noise levels. Moreover, the trend of misclassification rate as function of degrading SNR is studied for different thresholds that could be used as a measure of goodness of fit by the classifier. We again use a moving window of 0.1s over each of the test samples and output a predicted tap location using the kNN classifier based on maximum cross-correlation match.

The figures 4.5 and 4.7 show the misclassification error rate as function of background noise with varying SNR for the 1st dataset for tap locations (0,20), (10,0) and (-10,0), with threshold values set to 0.1 and 0.5 respectively. The curve with hollow circles show the misclassification rate when no accelerometer signal is used to detect a true tap event. The curve with stars show the error rate when classification is performed using the tier sensor fusion algorithm which detects a tap with accelerometer in the first tier and subsequently applies the kNN classifier on the audio signal if the conditions in first tier is met.
Figure 4.4: Threshold = 0.1

Figure 4.5: Misclassification rate as function of degrading SNR with talking in the background with threshold set to 0.1, for tap locations (0,20), (10,0) and (-10,0)

The error rate for the hollow circle plot corresponding to classification without using sensor fusion with threshold 0.1 in Figure 4.5 abruptly stops at SNR 0 because none the samples below 0 SNR get classified as they fail to meet the goodness of fit threshold, leading to 0 for both denominator and numerator in equation (4.1) for error rate calculation. Therefore, no tap event is detected or classified at those SNR level when the threshold is set to 0.1 for the tier classifier.
Figure 4.6: Threshold = 0.05

Figure 4.7: Misclassification rate as function of degrading SNR with talking in the background with threshold set to 0.05, for tap locations (0,20), (10,0) and (-10,0)

Figure 4.7 shows the error rate with and without sensor fusing for threshold 0.05 as the SNR degrades. The figures show that in the presence of background noise, as the SNR significantly degrades, the tier classifier using sensor fusion gives a noticeable performance improvement up to a very low SNR.

Figure 4.8 show the misclassification error rate as function of background noise with varying SNR for the second experiment containing tap locations (+10, 0), (+10,+10), (+15, 0), (+20, 0), (+20,+20), (+30,+30), (+5, 0). The results from this data set also demonstrates that the error rate is very large when only microphone is used for tap inference, as opposed to sensor fusion algorithm that is robust to noise even in the presence of increasing background noise.
Therefore, our proposed sensor fused solution for a robust tap localization solution is a tier classifier. The first tier identifies a tap event by running accelerometer detection over a moving window of 0.1s. This is done by first choosing a threshold for accelerometer z-axis based on the training data set, where it is set to be equal to the 20% of the minimum of the maximum amplitudes in accelerometer z-axis measured from all the template signals in the training phase. This threshold is then used to detect a true tap event during the detection phase. Once the first tier detects a tap using this criterion, a 0.1s window of microphone signal is captured which is passed onto the second tier. The second tier then applies the kNN classifier which uses cross-correlation analysis on the audio feature vector combined with a maximum voting scheme and the minimum goodness of fit requirement to output a predicted tap location for the tap sample. This corresponds to a tap detection response time of about $1/10^{th}$ of a second and results into an efficient and robust tap inference solution that can quickly train, and detect a tap.
with high accuracy even in the presence of various types of background acoustics and degrading SNR.
Chapter 5

Contiguous Tap Detection

In the last two chapters, we discussed a tiered algorithm combining accelerometer detection and a K-nearest neighbor classifier that can detect and classify discrete tap into tap locations with high success-rate even in the presence of noise. The algorithm discussed thus far can therefore classify a new tap into one of the tap locations whose templates was stored during the training phase. However, if we were to provide it with a new tap that is not in the database, such as a tap that is in between two template taps, the algorithm will output the location of one of the template locations. The desired output location would be the true position of the test tap. In this section, we focus our investigation on the problem of tap inference where the extended-touch can detect contiguous taps.

To study the inference of taps whose templates are not in the calibration database, we first start by analyzing how the maximum cross-correlation decays as a function of distance for any two different pair of tap templates from the calibration phase. Figure 5.1 shows the maximum cross-correlation value between each pair of templates for the knock tap data presented in Chapter 3 and 4 consisting of 17 different tap locations plotted as function of the distance between the templates. The figure shows a sharp drop and flattening of max-cross-correlation value as the distance between the two tap locations increases. The goal is to take this observation and investigate whether it is possible to approximate this relation using an exponential family function. However, it is also apparent from the figure that for a given distance, there is a range for the maximum cross-correlation value that is seen between two templates. Therefore, approximating the relation by an exponential function would capture the general trend, but will still not be able to account for the range of values seen for a given distance between a pair of tap locations.
In this section, we focus on trying to solve the tap inference in 1-D, which for the purpose of our investigation is the x-axis for the same data set of knock taps. As the first step, we try to infer location of a tap in between two templates using an exponential decay function. Using the same dataset of 17 tap locations, by only looking at the data points on the x-axis, and the simulation was trained on data points that are 5, 10 and 20 cm apart. The maximum cross-correlation values between each pair of the training tap samples was used to fit an exponential function and extract a decay constant. This function was then used to predict location of test taps that are in between two existing templates. Let $C_1$ and $C_2$ to be the two template tap locations, and there are $m$ number of training samples per class. The objective is to output location of a test tap $t_{test}$ whose template is not in the training set. The steps and functions used to predict the location of the tap in between two existing templates is described as follows:

$$X_m = \text{the maximum cross-correlation value between two taps}$$
\( X'_C = \text{average of } X_m \text{ relative to each training sample in class } C \)

\[
\Delta d = (-1/\alpha) \ast \log(X_m)
\]

\[
\alpha = 1/5 \text{ found in learning phase}
\]

\[
c_{1 frac} = \frac{X'^a_{C1}}{X'^a_{C1} + X'^a_{C2}}
\]

\[
c_{2 frac} = \frac{X'^a_{C2}}{X'^a_{C1} + X'^a_{C2}}
\]

\[
loc_{pred} = c_{1 frac} \ast (pos_{C1} + \Delta d_{C1})
+ c_{2 frac} \ast (pos_{C2} - \Delta d_{C2})
\] (5.1)

The delta distance \( \Delta d \) relative to each of the templates is scaled by the maximum cross-correlation value relative to the 2 templates and added to the template locations accordingly. The results of two trials are shown below:

- **Trial - Mid point prediction:**
  - Template location used: (10, 0), (20, 0)
  - Test location: (15,0)
  - Results: 4 out of 4 test samples were predicted accurately - 100% detection rate

- **Trial - In between point but not in the middle:**
  - Template location used: (-10, 0), (10, 0)
  - Test location: (-5,0)
  - Results: 4 out of 6 test samples were predicted accurately, giving a 67% detection rate.
  - The wrongly predicted ones are: -1.17 -1.82

So the results of above 2 preliminary trails show that if the training phase contains a set of 5, 10, 20 apart training taps, it may be possible to fit an exponential function using the training samples, and use the decay constant learned during the training phase to predict location of taps that are in between the training tap templates.

### 5.1.2 iPhone implementation and Results

Next, using the decay constant found from offline data analysis, a modified version of the algorithm discussed above was implemented on an iPhone. The new training phase consists of at least 2 templates of each training tap location denoted as tap class \( C_i \) with tagged location \((x_i, y_i)\). The detection phase uses accelerometer and microphone signals to detect a new tap using a modified version of the tiered sensor fusion algorithm as follows:
• compute the average maximum cross-correlation of \( t_{\text{test}} \) tap relative to each of the training class \( C_i \)

• use k-NN to output the 2 sorted closest neighbors, denoted here as \( C_{n1} \) and \( C_{n2} \) where \( n \) represents neighbour

• if the maximum cross-correlation values of \( t_{\text{test}} \) relative to each of the two neighbors are not similar within a pre-defined range, then pick the closest neighbor as the predicted location. Therefore, if \( \frac{X_{\alpha C_{n1}}}{X_{\alpha C_{n1}}} < \) pre-specified threshold which is set to 1.4 , output \((x_{n1}, y_{n1})\) as the predicted position

• if not, then use the following logic to predict the output location

\[
\begin{align*}
\Delta d_{C_{n1}} &= -5 \times \ln(X_{\alpha C_{n1}}) \\
\Delta d_{C_{n2}} &= -5 \times \ln(X_{\alpha C_{n2}}) \\
predX_{C_{n1}} &= [x_{n1} - \Delta d_{C_{n1}}, x_{n1} + \Delta d_{C_{n1}}] \\
predX_{C_{n2}} &= [x_{n2} - \Delta d_{C_{n2}}, x_{n1} + \Delta d_{C_{n2}}]
\end{align*}
\]

Then by looking at the differences between the four predicted locations, pick the values relative to \( C_{n1} \) and \( C_{n2} \) that results into the smallest difference. Then use equation (5.1) to output the final \( x \)-position. The results from preliminary testing of the real-time implementation in 1-D is about 60-70%.

### 5.2 Linear Regression - Training and Results

After having a reasonable success-rate with the preliminary algorithm, the next step was to come up with a regression algorithm that can be used on an iPhone to determine the value of the decay constant \( \alpha \) in the training phase. The goal is to reduce the number of training tap classes required to detect and predict tap locations. We start by going back to the maximum-cross-correlation to distance relationship for the knock tap data in x axis for the data set discussed in the previous section. Figure 5.2 plots the maximum cross-correlation values of various tap templates relative to other taps versus the distance between the pair of taps. Figure 5.2 shows that up to a distance of 20 cm, exponential function can approximate the trend well which no longer holds for distance of 30 cm. Moreover there is a non-linear behavior corresponding to the range the max-cross-correlation value which spans a range of values for the same distance on top of the
decay trend. Therefore, we focus on training a linear regression based on taps that are at +/-10 cm and +/-20 cm locations on x-axis, and only considering distances that are 10 cm and 20 cm apart. This can be used to predict tap locations within this distance range from two template tap locations. Therefore, the function we are trying to learn is:

\[ X_m = A \exp(\alpha \Delta d) \]
\[ \log(X_m) = \log(A) + \alpha \Delta d \]
\[ Y = \log(X_m), W^T = [W_0, W_1], W_0 = \log(A), W_1 = \alpha \text{ which becomes} \]
\[ Y = W^T \ast X \quad (5.2) \]

The optimum value of \( W^T \) in (5.2) can be learned by applying a gradient decent algorithm on the training data set. Therefore, using the same data set of knock taps, we now train a linear regression to determine the optimal \( W^T \). The training database now only consists of taps that are at 10 cm, and 20 cm away. The objective is to have a training phase that includes taps that are about 10 cm or 20 cm apart, and the tap detection algorithm can predict tap locations that are in between, example 5 cm from the closest training sample. Therefore the steps are:

- initialize \( W \) to a random number using matlab function \( W = 0.01 \ast \text{randn}(2,1) \)
- use learning rate = 0.001, regularizer = 50
- run simulation until converges
The optimal value of $W^T$ found for this data set was $W_0 = -0.0099, W_1 = -0.0851$.

Figure 5.3: Distance prediction error as function of distance to the template taps for test taps at (-10, 0) and (-15, 0)

The optimal $W^T$ found from training phase was used to predict distance for test tap samples on the x-axis using their maximum cross-correlation relative to each template tap as the input, which was then compared to their true distance from the template taps. The results are presented in figures 5.3, 5.4 and 5.5. The figures plot the error in distance predicted for four test tap locations relative to the template taps, where the absolute
value of the distance error is plotted against the distance to each of the templates for the test tap. The figures demonstrate that the distance error can vary a lot, but is
relatively small for distance of 0, 5 cm and 10 cm. Therefore the algorithm presented here could perform reasonably well to predict taps that are in between templates that are 10 cm apart.

Figure 5.6: Error distribution seen among the test templates for 0, and 5 cm away from the template taps.
Figure 5.7: Errors seen among the test templates for 10 cm, and 15 cm away from the template taps

Figures 5.6 and 5.7 shows the range of error seen in predicted tap locations among the test samples location at different truth distances relative the training templates. Using the values of $W^T$ found using gradient decent where $W_1 = -0.0851$, the inverse of the
Chapter 5. Contiguous Tap Detection

decay constant $\alpha$ is calculated to be $\frac{1}{\alpha} = 11.75$, which defers from the value found in last section Preliminary Results of $\frac{1}{\alpha} = 5$. The difference is that in this simulation, no training samples at 5 cm away was used, therefore, the algorithm performs better for distances that are 0, 10 cm and 20 cm away, and slightly poorly on taps that are 5 cm away.

If the training phase includes at least one tap location that is 5 cm away from another training tap, such an example training database could include taps at (10,0), (5, 0), and (-10,0), then the algorithm of tap inference should have better performance.

5.3 Future Research - Ensemble Method

Given the analysis for 1-D tap inference results, we propose an ensemble classifier to predict location of multiple taps as part of future research to solve for tap inference problem. The proposed classifier is a combination of the tier k-nearest neighbor classifier and the inference prediction using regression analysis. The algorithm is as follows:

- Training phase:
  - Collect tap samples that are about 10-20 cm apart, with at least one template tap at 5 cm away from others
  - Using the the training set, compute maximum cross-correlation value between each pair of tap templates
  - Use the maximum cross-correlation values along with corresponding distance between the pair taps $\Delta d$ to compute the decay constant $\alpha$ using regression analysis with gradient decent

- Detection phase:
  - Using K-nearest neighbor tier classifier discussed in Chapter 4, find two best training tap location match $C_{n1}$ and $C_{n2}$
  - If the maximum cross-correlation relative to either of top predicted classes $C_{n1}$ and $C_{n2}$ is higher than a set threshold, an example of such a threshold could be $0.7 \times \max(X_m)$ computed from the training set, output that class as the predicted location
  - If maximum cross-correlation relative to both class $C_{n1}$ and $C_{n2}$ is similar, and $C_1 \neq C_2$, use the decay constant $\alpha$ and equation (5.1) to output the predicted location relative to $C_{n1}$ and $C_{n2}$ in 2-D space.
Due to time constraint, this algorithm was only implemented in 1-D, not in 2-D, but one could easily extend the 1-D solution to output a 2-D predicted location for the test tap. This is left to be tested as a future investigation towards solving for contiguous tap detection and classification.
Chapter 6

Conclusion

In this thesis, we presented an algorithm that detects discrete tap locations by applying cross-correlation on the impulse response captured by the microphone on iOS platform. We demonstrated that by using simple machine learning algorithm with a training phase, we were able to develop a portable single sensor solution that works on various surfaces. Our proposed solution had a high success-rate for hard taps, while yielding a lower detection rate for soft taps. We also found that soft taps are not correlated with hard taps at a given location, nor are they confused with hard taps from other locations, alluding to different modes of wave propagation from different strength of taps. However, if the training phase includes both the soft and hard taps, subsequent taps can be detected with fairly high accuracy.

Following the satisfactory results from the kNN classifier, we improved upon the detection by proposing a tiered sensor fusion algorithm that uses accelerometer and microphone on iOS platform to detect and localize taps with 100% detection rate (for the data sets studied in this paper) even in the presence of people conversing in the background. Moreover, it was demonstrated that by fusing these two sensors the algorithm not only results into high detection rate even in the presence of background acoustics, it also becomes robust to degrading SNR of up to 10dB or even lower. We demonstrated that by using simple machine learning algorithm with a training phase, a portable solution is achieved that works on any surface where the device is placed. The tap detection algorithm and sensor fusion results are also discussed is [29, 30].

Finally, we studied a method to solve for contiguous tap detection - an ensemble method which combines the tier classifier with KNN detection as well as regression analysis to approximate a function that can be used to predict the location of a tap that is near the template taps. The algorithm was implemented and tested in 1-D, which gave better results if the template taps are within 10-20cm of the test tap. The ensemble
algorithm implemented in 1-D can easily be extended to solve for 2-D location of a tap on the surface, which is left for future investigation of extended touch solution.

Extended touch technology could be used to define new ways of interacting with our smartphones that allow for a natural interface for various use cases. One such example is developing a gaming interface or a musical instrument that use taps at different locations on the surface as input, which can easily be extended to multi-player platform. One could also utilize the detection capability to map various user defined actions such as sending an email, flipping through pages or snoozing the alarm, to different tap locations. The technology can also be extended to detect other types of taps such as jumping on the floor which can be detected to develop a work-out or dance detection system. The robustness of the algorithm to degrading noise level and background sound makes such extended touch applications more usable and feasible user interface. Live videos demonstrating some of the applications of extended touch technology discussed above could be viewed at [1].

Future research step for this problem would be to recognize contiguous tap locations in 2-D where not all tap locations have pre-existing templates in the training database. Moreover, the success rate can decrease if the taps are not consistent between the training phase and detection phase. The current real-time implementation of extended touch on the iOS platform does not combine more than two templates per tap output classes. As an extension of current work, one can implement any of the multiple-template combining methods discussed in Chapter 4 to boost the performance of the detection algorithm and make it robust to tap inconsistency.
Bibliography


