Improving Pedestrian Safety in Cities using Intelligent Wearable Systems

Stephen Xia, Daniel de Godoy, Bashima Islam, Md Tamzeed Islam,
Shahriar Nirjon, Peter R. Kinget and Xiaofan Jiang

***Abstract*—With the prevalence of smartphones, pedestrians and joggers today often walk or run while listening to music. Since they are deprived of their auditory senses that would have provided important cues to dangers, they are at a much greater risk of being hit by cars or other vehicles. In this paper, we build a wearable system that uses multi-channel audio sensors embedded in a headset to help detect and locate cars from their honks, engine and tire noises, and warn pedestrians of imminent dangers of approaching cars. We demonstrate that using a segmented architecture consisting of headset-mounted audio sensors, a front-end hardware platform that performs signal processing and feature extraction, and machine learning based classification on a smartphone, we are able to provide early danger detection in real-time, from up to 60m away, and alert the user with low latency and high accuracy. To further reduce power consumption of the battery-powered wearable headset, we implement a custom-designed integrated circuit that is able to compute delays between multiple channels of audio with nW power consumption. A regression-based method for sound source localization, AvPR, is proposed and used in combination with the IC to improve the granularity and robustness of localization.**

***Index Terms*—wearables, sound source localization, pedestrian safety, embedded systems.**

I. Introduction

Smartphones have transformed our lifestyles dramatically, mostly for the better. Unfortunately, smartphone usage while walking has become a serious safety problem for many people in urban areas around the world. Pedestrians listening to music, texting, talking or otherwise absorbed in their phones are putting themselves at risk by tuning out the traffic around them [2], as reported by the Washington Post. Since a pedestrian is deprived of auditory input that would have provided important cues to dangers such as honks or noises from approaching cars, he or she is at a much greater risk of being involved in a traffic accident. We have seen a sharp increase in injuries and deaths from such incidents in recent years. According to a study by Injury Prevention and CNN, the number of serious injuries and deaths occurring to pedestrians who were walking with headphones has tripled in the last seven years in the United States [3]. This phenomenon affects cities globally, and is an important societal problem that we want to address by introducing advanced sensing techniques and intelligent wearable systems.

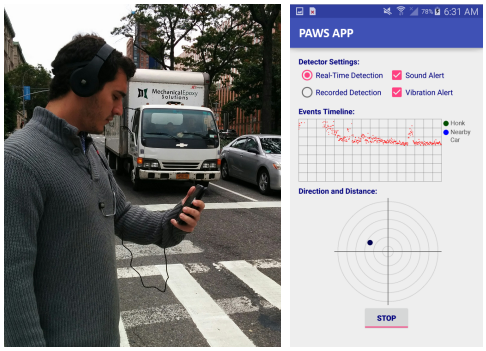


Fig. 1. An inattentive pedestrian wearing a PAWS headset, and a screen shot of the PAWS application user interface.

We tackle these challenges in PAWS, a *Pedestrian Audio Wearable System* aimed for urban safety. PAWS is a low-cost headset-based wearable platform that combines four MEMS microphones, signal processing and feature extraction electronics, and machine learning classifiers running on a smartphone to help *detect* and *locate* imminent dangers, such as approaching cars, and *warn* pedestrians in real-time. Figure 1 shows PAWS in action.

With newer smartphones equipped with multiple built-in microphones, it may be tempting to re-purpose those microphones in software to localize cars based on common localization techniques. However, these approaches require the user to constantly hold their phones steady and to not block the built-in microphone while walking [4][5]. Further, most built-in microphones are designed for voice and are often band-limited. These two limitations prevent the smartphone from capturing useful features produced by approaching cars in realistic urban environments.

This is a challenging problem as the battery-powered wearable platform needs to detect, identify, and localize approaching cars in real-time, process and compute large amounts of data in an energy and resource constrained system, and produce accurate results with minimal false positives and false negatives. For example, if a user's reaction-time is 500ms, the system has 360ms to detect a 25mph car and alert the user when it is 10m away. This problem is further compounded by high levels of mixed noise, typical of realistic street conditions.

We address these challenges over two phases of design

S. Xia, D. de Godoy, P. Kinget and X. Jiang are with the Department of Electrical Engineering, Columbia University, New York, NY, 10027 USA, email: {stephen.xia, dd2697, peter.kinget, xiaofan.jiang}@columbia.edu.

B. Islam, M. T. Islam and S. Nirjon are with the Department of Electrical and Computer Engineering, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599 USA, email: {bashima, tamzeed, nirjon}@cs.unc.edu

Parts of this work have been previously published in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)* [1].

and development. In the first phase (PAWS), we develop a segmented architecture and data processing pipeline that partitions computation into processing modules across a front-end hardware platform and a smartphone. The microcontroller-based front-end hardware platform consists of commercial off-the-shelf (COTS) components embedded into a standard headset and collects four channels of audio from four MEMS microphones that are strategically positioned on the headset. Temporal-spatial features such as relative delay, relative power, and zero-crossing rate are computed inside the front-end platform using the four channels and transmitted wirelessly to a smartphone. A fifth standard headset microphone is also connected to the audio input of the smartphone, and together with the data sent from the front-end platform, classifiers are trained and used to detect an approaching car and estimate its azimuth and distance from the user.

In the second phase of development (PAWS Low-Energy), we tackle the challenge of power consumption through the design and implementation of an application-specific integrated circuit to extract some of the computationally expensive features. We also develop new methods to increase the accuracy and granularity of our audio-based vehicle localization. We evaluate PAWS using both controlled experiments inside parking lots and real-world deployments on urban streets.

We make the following contributions in this paper:

- We propose a new acoustic feature, **Non-Uniform Binned Integral Periodogram**, which is designed to capture frequency domain characteristics of low-frequency noise-like sounds, such as the sound produced by the friction between a car's tires and the road. We develop classifiers to recognize cars approaching the user and to localize approaching cars in real-time.
- We create, **PAWS**, a low-cost, end-to-end wearable system using COTS components accompanied with a smartphone application to provide real-time alerts of oncoming cars to pedestrians in noisy urban environments. We demonstrate that inattentive pedestrians can immediately benefit from our system.
- We present a second system, **PAWS Low-Energy**, that improves upon the power consumption of our COTS implementation by offloading critical and computationally expensive features onto an application-specific integrated circuit. We additionally introduce **Angle via Polygonal Regression** (AvPR), an easily calibrated method for estimating the direction of arrival of car sounds. AvPR improves upon the granularity of direction estimates over the classification approach employed in PAWS and accommodates for noise better than classical geometric approaches (e.g. triangulation) for estimating direction, while remaining computationally inexpensive.
- We develop a segmented architecture and data processing pipeline that intelligently partitions tasks across the front-end hardware and the smartphone and ensures accuracy while minimizing latency.

As the industry is investing heavily in intelligent headphones [6][7], our hardware-software co-design approach presents a compelling solution towards protecting distracted pedestrians in urban environments.

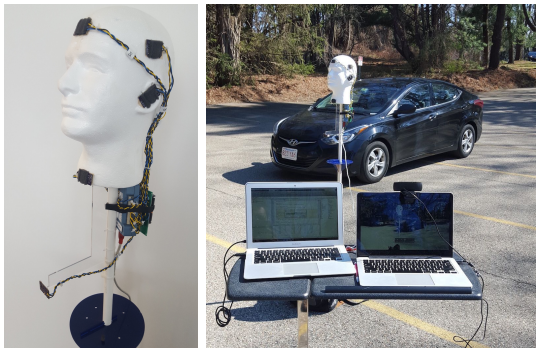


Fig. 2. Validation system: reference mannequin with eight MEMS microphones and data acquisition board in a low-noise experimental setup.

II. Studying the Problem

Before developing PAWS into a wearable system, we studied the car sound recognition and localization problem using a validation platform. The objective of this exercise was to analyze the feasibility and complexity of our proposed solution and to determine the specifications required to capture the necessary information, e.g., audio sampling rate, sensor placement, and most relevant features to use in the machine learning algorithms.

As shown in Figure 2, the platform directly connects eight MEMS microphones to a computer. The microphones were placed on a mannequin head to reproduce the physical phenomena of the final setup, such as the acoustic shadow of the human head [8] and the approximate spacing among sensors on a real user.

The study has been done in five different locations in two different cites: a metropolitan area and a college town. The locations were two parking spaces, a four-way intersection, and two multi-lane streets. We analyzed recorded audio from 47 different cars. Other than the parking spaces, where we conducted our first set of controlled experiments with labeled distances, directions, and precise time-keeping of honks and car passing, all other scenarios were uncontrolled.

A. Recording Specifications

In order to characterize the sounds of interest, such as an approaching vehicle's tire friction, engine noise, and honks, we conducted controlled experiments in two parking lots (Figure 2 shows one of the experiments). These results are later cross-checked against uncontrolled experiments' for consistency.

Figure 3 shows the spectrogram of one of the recordings from the controlled experiments. Both the top and the bottom figures correspond to the same recording. Approximately 5s after the recording starts, a car honks, resulting in distinct stationary tones with fundamental frequencies near 500Hz. The vehicle then accelerates towards the mannequin. In the bottom figure, where the lower part of the spectrogram is highlighted, we see the engine noise. The engine noise follows its RPM. In an automatic car, the engine noise is bounded between 60Hz and 200Hz (at the 7 seconds mark, the shift in the engine gear is noticeable). Once the vehicle gets closer

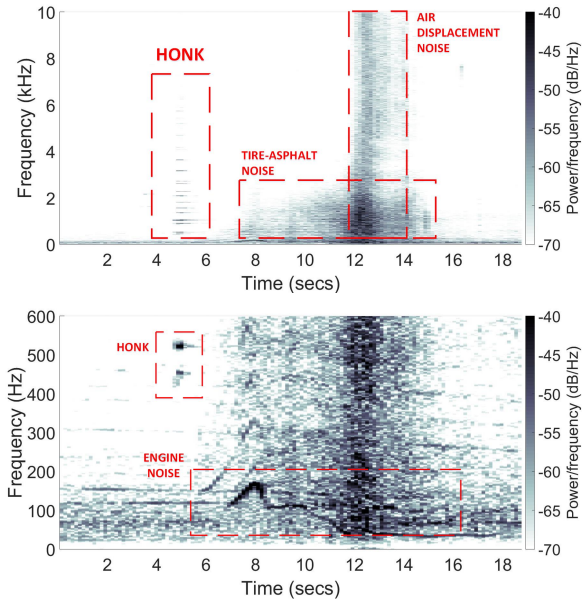


Fig. 3. Spectrogram of one of the recordings from the controlled environment. The car was approaching the mannequin at 25mph.

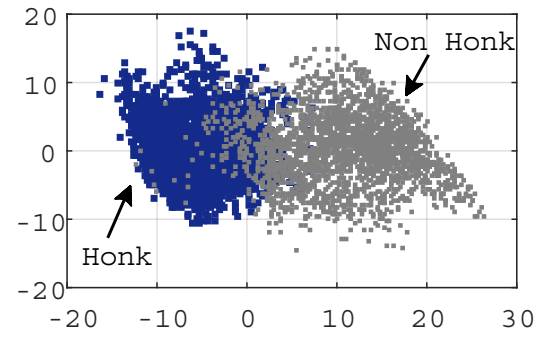


Fig. 4. Distribution of honks and other types of sounds in a 2D feature space.

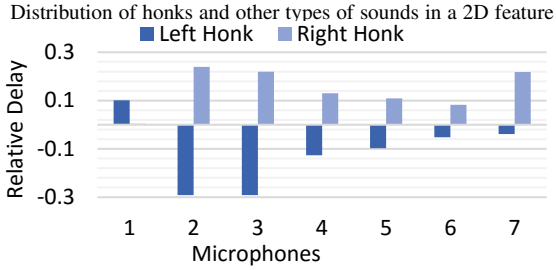


Fig. 5. Normalized relative delays of the microphones for left and right honks.

to the mannequin, the friction noise from the tires and asphalt gets louder. This noise has a band-limited spectrum with more energy below 3kHz. When the car crosses the system near the 12s mark, a burst of air causes a loud white noise. Similar spectrum components were found on several recordings of different cars at similar speeds (20-30mph) on dry asphalt.

These observations indicate that to identify warning honks and vehicles that are still approaching the user, the system audio must reliably capture frequencies from 50Hz to 6kHz. This requirement means that the system needs custom microphone drivers with a cut-off frequency of less than 10Hz (in contrast to standard headset microphones with approximately 100Hz cut-off frequency) and analog-to-digital converters with sampling rates above 12kSamples/s.

B. Presence of a Car

The presence of a car can be determined from high-energy, sharp sounds like honks, as well as from low-energy, noise-like sounds such as the sound of friction between a tire and the road. Being able to detect cars based on friction noise is crucial given the increasing popularity of electric cars with quieter engines.

Honks are louder and, thus, easier to detect than car tire or engine sounds. We analyze the Mel-Frequency Cepstral Coefficients (MFCC) [9] of honks and compare them with non-honk street sounds. We start with MFCC, since it is one of the most commonly used acoustic features for detecting various types of sounds [10][11][12][13] including car sounds [14]. For visualization purpose, we reduce the 13-dimension MFCC features to two dimensions (using PCA [15]) and the result is shown in Figure 4. We observe that honks are separable from other sounds as they cluster around a different point in space. Honks are easily detectable using all 13 coefficients.

MFCCs, however, are not effective in detecting other types of car noises such friction between tires and the road. The fundamental reason behind this is that the Mel-scale, expressed by $m = 2595 \log_{10}(1 + f/1000)$, was originally designed to mimic human hearing of speech signals that maps frequencies $f <$ 1kHz somewhat linearly, and maps $f >$ 1kHz logarithmically. Our analysis on tire friction sounds shows that about 60% signal energy is attributed to frequency components below 1 kHz. Hence, to model such low-energy, low-frequency, noise-like sounds, we need to develop a new feature that captures these sub-kHz characteristics. We propose this new feature in Section III-C1.

C. Direction of a Car

To determine the direction, we record audio of cars approaching from different directions and analyze their effect on the microphone set. Some of these recordings also have honks in them. Intuitively, microphones that are closer to the sound source and are not obstructed by the human head should receive signals earlier, and the signals should be stronger. Hence, the relative delays and the relative energy of the received signals should be strong indicators of the direction of an approaching car.

In Figure 5, we plot the relative delays of the microphones with respect to the front microphone for left and right side honks. We see that the relative delays change signs for left and right honks. We do similar tests with eight directions (each covering a 22.5^o 3D cone surrounding the mannequin) to successfully determine the directions of honks near the user.

Similarly, we plot the relative delays of the microphones for a car that passes the mannequin from its left to the right (Figure 6). We observe that the relative delays are quite random on both left and right ends. As the car approaches the

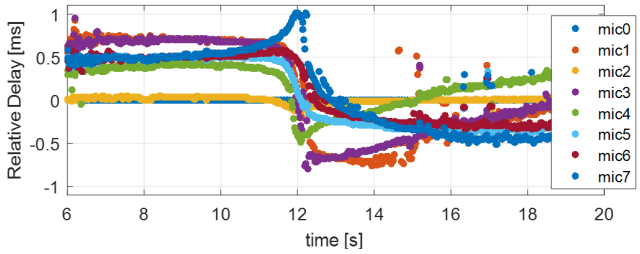


Fig. 6. Relative delay versus time of a car driving past a mannequin from its left to right.

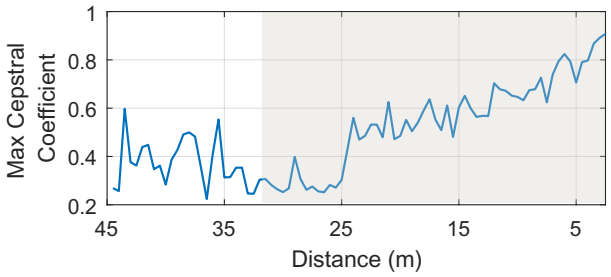


Fig. 7. The maximum cepstral coefficient follows a trend when an approaching car is within about 30m from an observer.

mannequin, we see a trend in all the curves with one or more of them reaching their peaks. The trend reverses as the car passes the mannequin. This behavior suggest that patterns in relative delays (when they are looked at together) are useful to determine the direction of passing. Hence, by learning the trend and the point when the trend reverses, it is possible to differentiate between a car on the left from a car on the right, as well as their angular directions.

D. Distance of a Car

In an attempt to estimate the distance, we formulate a regression problem that maps sound energy to distances. Later we realize that due to environmental noise and the weakness of car sounds, a fine grained location estimation is extremely inaccurate when the car is farther than 30m from the audio recorder. When the car is within 30m, we find that the maximum value of the cepstral coefficients (computed every 100 ms) is approximately linearly correlated with distance, as shown in Figure 7 for a car that is driven toward the mannequin. This relationship can be exploited to form a regression problem that maps maximum cepstral coefficients to distances.

For cars farther than 30m, although we are able to detect their presence and estimate their direction, a precise distance estimation results in a large error. However, we learn that the distance estimation problem can be formulated as a multi-class classification task by dividing the absolute distances into a number of ranges such as (0, 20m], (20m, 40m], and (40m, 60m]. Each of these ranges can be characterized with additional features, such as zero-crossing rate, and can be classified accurately using a machine learning classifier.

As such, one option is to use a two-level approach for distance estimation. The first level employs a classifier to

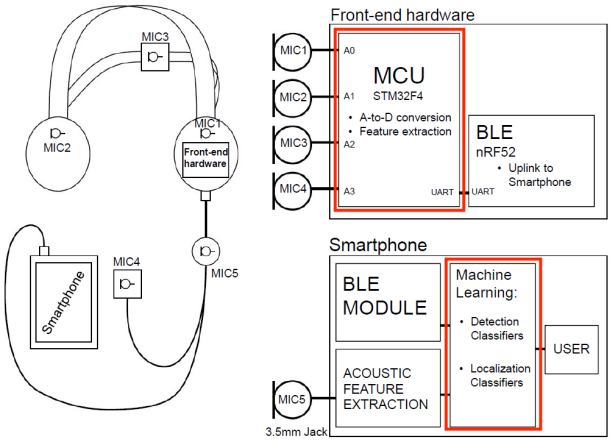


Fig. 8. A block diagram of PAWS. The components highlighted in red are portions of the system that are enhanced and modified during the second phase of development to create PAWS Low-Energy, which is discussed in Section IV.

determine a coarse-grained distance range, and if a car is detected within the nearest range, it applies regression to obtain a fine grained distance estimate. A second option is to disregard estimating coarse-grained distance and only provide fine-grained distance measurements via regression. As shown in Figure 7, for distances above 30 meters, the maximal cepstral coefficient maps to approximately the same value. As such, the regression model inherently classifies car distances into two coarse groups: greater than 30 meters and less than 30 meters away. We considered both methods over the course of designing both phases of PAWS, as detailed in Sections III and IV.

III. Overview of PAWS

PAWS is a wearable headset platform and smartphone application that uses five microphones and a set of machine learning classifiers to detect, identify, and localize approaching cars in real-time and alerts the user using audio/visual feedback on his smartphone.

The system consists of three main components: sensors and their drivers, front-end hardware for multi-channel audio feature extraction, and a smartphone host for machine-learning based vehicle detection and localization, which are shown in Figure 8. Four of the MEMS microphones, labeled MIC1 to MIC4, are distributed over the user, at the left and right ear, back of the head, and chest of the user, to provide relevant information about the sound source's location. The front-end hardware synchronously acquires analog signals from these microphones and locally extracts acoustic features that are used by a smartphone application. PAWS performs signal processing inside the front-end hardware so that only features need to be transmitted wirelessly to the smartphone (via BLE) instead of large amounts of raw audio data. The front-end hardware is a battery-powered embedded platform that is housed within the confines of the headset that uses its own set of microphones for sound processing. It does not interact with the speakers or microphone of the headset. As such, a user would have not experience any degradation in sound or microphone quality of the headset.

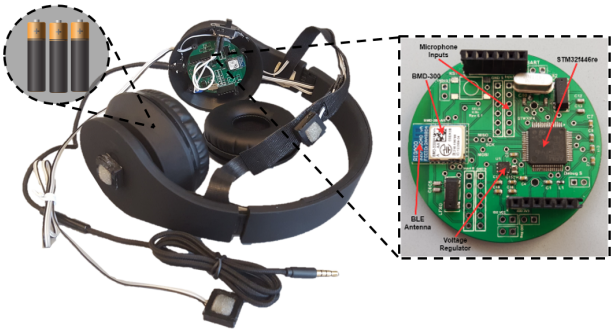


Fig. 9. (Left) Teardown of the PAWS headset; the front-end hardware is exposed inside the left ear housing. (Right) Close up of the PCB that comprises the PAWSfront-end hardware.

The standard microphone of the headset (the fifth microphone, MIC5) is connected to the 3.5mm audio input of the phone. Data from the fifth microphone is directly acquired by the smartphone. The audio from the smartphone/headset microphone is acquired in the same way as common messaging and calling applications and does not affect the quality or user experience of the microphone. Using the features computed by the front-end hardware and an audio stream from the headset microphone as inputs, machine learning classifiers running inside the PAWS application detects the presence of an approaching vehicle and estimates its position relative to the user. Our architecture uses a single low-power microcontroller in the front end and relies on the smartphone to run machine-learning classifiers to deliver reasonable latency.

A. Front-End Hardware

The front-end hardware is responsible for three blocks on the PAWS signal flow: synchronous ADC of microphone channels, embedded signal processing, and wireless communication with the smartphone. The integration of these blocks in a wearable resource-constrained system is a challenging task, and computational bottlenecks such as memory and data transfer rate require a careful distribution of resources.

In order to demonstrate PAWS's system architecture and algorithms, off-the-shelf components were used to build the system. As shown in Figure 8, four MEMS microphones are wired to an MCU. The MCU synchronously collects the signals, calculates the temporal-spatial features, and sends the result to a smart BLE module via UART. The BLE module sets the link between the front-end hardware and the smartphone. The front-end hardware is powered by standard AAA batteries and is designed to fit inside the left ear housing of a commercial headset, as shown in the left figure in Figure 9.

B. Front-End Signal Processing

In this section we discuss the operations that are processed by the front-end hardware. The MCU must sample the data from the four MEMS microphones and perform feature extraction, while the BLE module is responsible for transferring the calculated features to the smartphone. Since cars may be traveling at high speeds, fast response times and low latency are critical. PAWS uses a Cortex-M4 MCU to perform data acquisition and processing in real-time. The design choices and evaluation are explained in detail in Section V.

1) Sampling Data: Audio is captured from four microphones at 32kSamples/s with an 8-bit successive approximation ADC and a four channel analog multiplexer running in the microcontoller. The sampling frequency was chosen as a compromise between the lowest rate necessary to capture the spectral content, as explained in Section II, and the performance enhancement achieved by a delay estimation with finer granularity.

2) Feature Extraction: Running the feature extraction algorithms in real-time in a Cortex-M4 is challenging due to the complexity and number of computations required across the four channels. In order to service a continuous stream of incoming data, it is imperative that the feature extraction finishes before the next window of data is completely received. The feature extraction calculations were simplified to achieve low latency; complex multiplications and division were avoided. The following features were calculated on the acquired four channels of data: relative power of each channel with respect to MIC1, relative delay with respect to MIC1, and zero-crossing rate of each channel. These features are calculated for every time window of 100ms with 50% window overlap.

The relative power ($Rp_{N,1}$) is calculated by summing the difference of squares between samples from each microphone to the reference microphone, MIC1.

$$\mathrm{Rp_{N,1}} = \sum_{\mathrm{i=1}}^{\mathrm{W_L}} (\mathrm{X_N^2[i] - X_1^2[i]}) \tag{1}$$

N is the channel number, W_L is the window length (in this case 3200 samples), X_N is the channel signal, and X_1 is the reference MIC1 signal.

The relative delay is calculated using cross-correlation. The lag between the channels is defined as the index where the cross-correlation ($\mathrm{XCORR_{N,1}}$) is maximum.

$$\mathrm{XCORR_{N,1}[d]} = \sum_{\mathrm{i=0}}^{\mathrm{W_L}} \mathrm{X_N[i-d].X_1[i]} \tag{2}$$

This is the most computationally expensive calculation of the front-end system. Since the physical separations of microphones are limited, e.g. the average spacing between ears is ∼25cm, the range of valid relative delay is bounded, making it possible to compute and compare the XCORR only for $\mathrm{d} \in [-40, 40]$. According to [16], these limits on the time interval of interest make computing cross correlation in the time domain much more efficient than computing in the frequency domain.

The zero-crossing rate ($\mathrm{ZC_N}$) is the number of times a signal changes sign within a given time window.

$$\mathrm{ZC_N} = \sum_{\mathrm{i=1}}^{\mathrm{W_L}} (|\mathrm{sgn(X_N[i])} - \mathrm{sgn(X_N[i-1])}|) \tag{3}$$

3) Data Transfer: The BLE module gathers the resultant 10-element feature values and sends them to the smartphone following a custom protocol in 40 byte packets. The protocol consists of a validation header (3 bytes), followed by a set of

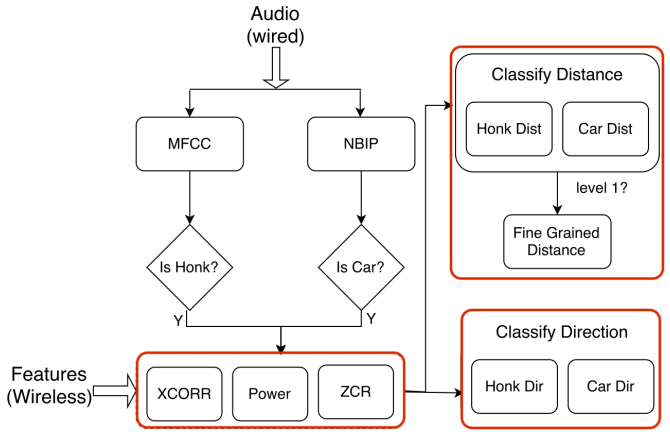


Fig. 10. PAWS Smartphone data processing. The components highlighted in red are portions of the system that are improved upon in PAWS Low-Energy, which is discussed in Section IV.

hardware configuration flags (1 byte), payload size (1 byte), and the feature values (1×3 bytes for relative delays of MIC $\{2, 3, 4\}$, 8×3 bytes for relative powers of MIC $\{2, 3, 4\}$, and 2×4 bytes for ZC of all four microphones).

C. Smartphone Data Processing

The PAWS smartphone app receives a 44.1kHz, single channel audio stream from the headset via the standard microphone jack, acoustic features over BLE from the front end, and processes them in real-time in a service. The application comes with a graphical user interface that is used to start/stop the service, configure alerts, and display a timeline of approaching cars along with their distances and directions.

Figure 10 shows the data processing pipeline of the PAWS smartphone application. The application implements a two-stage pipeline for detecting and localizing cars, respectively.

1) Car Detection Stage: Two offline-trained classifiers are used in this stage to detect cars honks and engine/tire sounds. The first classifier uses standard MFCC features to detect the presence of car honks. For the other type of car noises, we propose a new acoustic feature, *Non-Uniform Binned Integral Periodogram (NBIP)*, that unequally divides the frequency scale in order to capture variation in spectral energy at the lower end of the frequency spectrum which characterizes the friction sound from car noises. The steps to compute the NBIP features are as follows.

- **Step 1:** The FFT of each audio frame $x(t)$ is computed to obtain the Fourier spectra $X(f)$. Only the left half of this symmetric spectra is retained.
- **Step 2:** The periodogram of $x(t)$ is obtained from $X(f)$ by normalizing its magnitude squared, and then taking its logarithm.

$$P_x(f) = 20 \log_{10} \left(\frac{1}{F_s N} |X(f)|^2 \right)$$

F_s and N denote the sampling frequency and the signal length, respectively.

- **Step 3:** The frequency range is divided into a total of B bins, such that the frequencies below a threshold a are equally divided into b bins, and the higher frequencies are equally divided into $B - b$ bins. The binning process is illustrated in Figure 11. The optimal values of the parameters B, a, and b are empirically determined, which we will describe shortly.

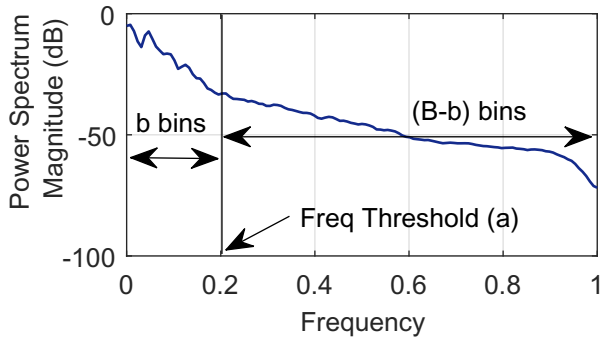


Fig. 11. Illustration of the basic idea of non-uniform binning of spectral energy in NBIP.

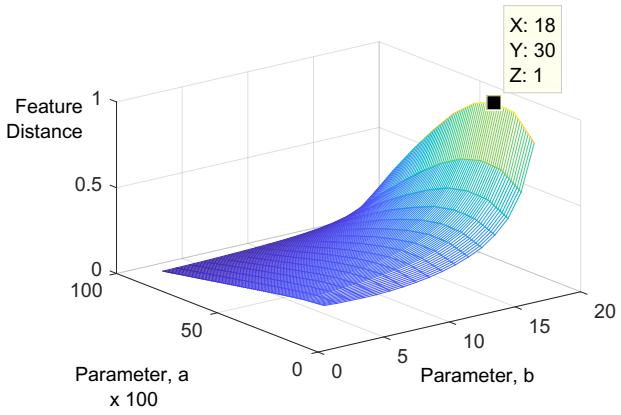


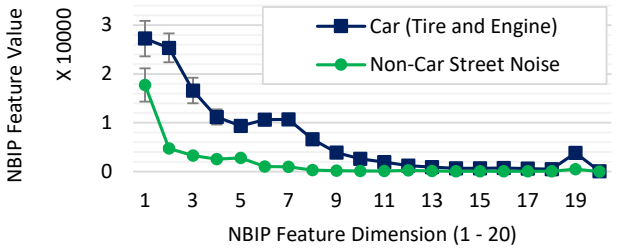
Fig. 12. NBIP search space for parameter optimization.

- **Step 4:** The $P_x(f)$ is integrated in each bin to obtain a B dimension feature vector $v = (v_1, v_2, \ldots, v_B)$.

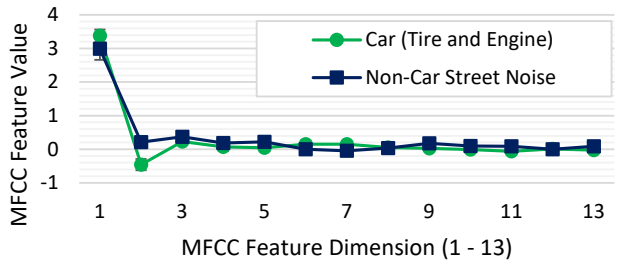
$$v_k = \begin{cases} \int_{(k-1)\Delta_1}^{k\Delta_1} P_x(f)df, & \text{if } 1 \le k \le b \\ \int_{a+(k-b-1)\Delta_2}^{a+(k-b)\Delta_2} P_x(f)df, & \text{otherwise} \end{cases}$$

where, $\Delta_1 = \frac{a}{b}$ and $\Delta_2 = \frac{1-a}{B-b}$ are the bin sizes for frequencies below and above the threshold a, respectively.

In order to find the optimum values of the parameters a, and b, we vary the parameters $0 \le a \le 1$ and $1 \le b \le B$ in small increments and compute the vector difference between features of car noises and all other non-car sounds. Figure 12 shows the search space for a and b for a fixed value of $B = 20$. We observe that when $a = 0.3$ and $b = 18$, the vector difference between the car noise features and the non-car sound features is maximized. Figure 13 shows the mean and standard deviation of each component of the two types of feature vectors (i.e. NBIP and MFCC), for the two classes of sounds. We observe that most of the NBIP feature components (e.g., the first 10 components) are very dissimilar for the two



(a) NBIP feature vector for car and non-car sounds.



(b) MFCC feature vector for car and non-car sounds.

Fig. 13. (a) The proposed NBIP feature vector for car tire and engine sounds are designed to maximize their dissimilarity from non-car street noises like human chatter, human motions, machine sounds, and loud music. (b) standard MFCC features are not as effective in separating the two classes as our proposed NBIP features.

classes, whereas the MFCC features for both classes are very similar. Unlike MFCCs, NBIPs are designed to maximize their vector representations for car engine/tire vs. non-car sounds, which makes them effective in recognizing cars with a very high accuracy.

The NBIP features introduced are only used to detect approaching cars/engine and tire noises. Since honks exhibit strong frequencies in narrow bands and are not noise-like, we cannot use NBIP to accurately detect honks. As such, we use standard MFCC features for honk detection. For both types of classification (honks vs. engine/tire noises), we train separate Random Forest classifiers [17] which perform significantly better than other classifiers (e.g., Support Vector Machine [18]) that we applied on our data set.

The classifiers were trained using audio recorded from 60 different vehicles, ranging from sedans to buses and trucks, including the initial 47 recorded for studying the problem. We found similar performance using classifiers trained with sounds recorded from as low as 30 different cars. Additionally, we included environment sounds without cars recorded from the college town and metropolitan areas in which we conducted our study. These audio clips include a wide range of non-car sounds typically found in an outdoor environment, including but not limited to talking, wind, and wildlife.

2) Car Localization Stage: If the presence of a car is detected, the second stage of the pipeline is executed. In this stage, the smartphone acquires and uses the four-channel acoustic features received from the embedded front-end system to estimate the distance and direction of the car. Four multi-class Random Forest classifiers are used to classify eight directions and three distance levels based on honks and engine/tire-friction sounds, respectively. Because the feature vectors are only of 10 dimensions, we feed all the features into both

PERFORMANCE SUMARY	
Technology	180nm CMOS
Sampling Frequency	50KS/s
Range	[-2.52ms,2.52ms]
T_{LSB}	20µs
Peak INL	-1.57/1.33 LSB
Peak DNL	-0.85/0.97 LSB
TDE ENOB	6.06bits
Number of TDE Channels	3
Comparator Power (Unit)	3.1nW
PCC-ATDE Core Power	65.9nW
Total Power	78.2nW
TDE Energy per Conversion Step per Channel	7.84fJ/Conv.-Step
Active Area	0.28mm²
Die Area	1mm²

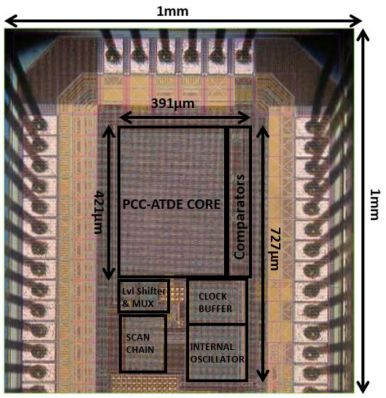


Fig. 14. Custom integrated circuit for computing acoustic features directly from multi-channel audio [19], used in PAWS Low-Energy.

classifiers for a simpler implementation. However, our analysis of principal components (PCA) reveals that relative delay and relative powers are more relevant features for direction classification, whereas relative delay combined with ZC and relative power are relevant features for distance estimation. Relative delay is relevant to the direction of the sound source because the microphone closer to the sound source will receive the audio signal sooner than the other microphones.

In addition to determining one of the three levels of distances, when a car is detected within the nearest level (within 30m), PAWS runs a linear regression-based fine-grained distance estimator. This step includes computing the cepstral coefficients and then fitting the maximum value to an actual distance in meters. This step does not add any significant cost as we obtain the cepstral coefficients as a byproduct of MFCC computation during the car detection stage.

3) Alert Mechanism: The application alerts a user with audio/visual feedback. If a car is detected within a user-configured distance range (e.g., 40m) – the phone vibrates, lowers the volume, and beeps. It can also be configured to play a customized message, e.g., "a car is {approaching, honking} on your {direction, left, right}". The application also visually shows the location and direction of the car on its user interface, as shown in Figure 1.

IV. PAWS Low-Energy

The PAWS front-end platform uses an MCU to sample and compute audio features used in direction and distance classification. Some of these features, such as cross-correlation, are computationally expensive and grow quadratically in window size. While the MCU-based sensing system we developed for PAWS is already optimized for power consumption, it still consumes significant energy since MCUs are general-purpose processing units that are not optimized for our specific application. To address this challenge, we further reduce power consumption of the front-end platform by designing and implementing an application-specific integrated circuit (ASIC) to compute digital acoustic features directly from analog sound sources, as shown in Figure 14. In this section, we introduce PAWS Low-Energy, an improved version of PAWS, that uses our custom ASIC in place of an MCU to sample

audio and compute critical audio features for localization to further reduce power consumption. Additionally, we discuss the limitations of using the ASIC and how our localization methods change as a result. Finally, we introduce our computationally efficient, noise-resilient, mapping-based method for direction localization, Angle via Polygonal Regression (AvPR), that provides finer granularity direction estimates than the eight directions provided by the classifier-based method implemented in PAWS.

A. Reducing Power through a Custom Application-Specific Integrated Circuit

To reduce the power consumption of acoustic feature computation, we replace the MCU in the front-end platform with a custom-designed ASIC. We describe the design, fabrication, and evaluation of our ASIC in detail in [19]. This ASIC computes the relative delay between three channels of audio with respect to one channel of audio in the analog domain, using a technique called polarity-coincidence correlation. The polarity-coincidence correlation (PCC) between two signals, X_1 and X_2, is shown in Equation 4 [20].

$$\mathrm{PCC}_{\mathrm{X_1,X_2}}(\tau) = \int_{\mathrm{t}}^{\mathrm{t}+\tau} \mathrm{sgn}(\mathrm{X_1(t)}).\mathrm{sgn}(\mathrm{X_2(t-\tau)})\mathrm{dt} \quad (4)$$

The main difference between standard cross-correlation and PCC is that PCC computes the relative delay between two signals using only the the signs of signals instead of the entire signals. Because only the signs of the signals are required, our custom ASIC does not need to sample entire audio streams to compute relative delay and is able to efficiently extract the relative delay through a feedback circuit using only comparators for computing signs and memory elements for shifting signals. This design leads to a reduction in power consumption of the front-end to nW levels compared to the mW level consumption by using ADCs on an MCU to directly sample the audio streams. Using the custom IC, we show in Section V-B that the majority of consumption is due to other components of the front-end, such as the BLE module. This reduces the overall power consumption of the wearable system by an order of magnitude, allowing us to power PAWS Low-Energy for longer than PAWS even after replacing the AAA batteries with CR2032 coin cells, which have one order of magnitude less energy capacity.

B. Resolving Constraints Imposed by Utilizing the ASIC

As mentioned in Section III-B, PAWS uses a variety of features, including zero-crossing rate and relative power, for direction and distance classification. However, the ASIC only provides a relative delay measurement; thus, PAWS Low-Energy cannot utilize the same set of methods used in PAWS.

The distance classifier introduced in Section III-C uses relative power and zero-crossing rate, two features not computed by the ASIC, to classify the distance of a detected car into three coarse ranges. If the classifier detects that a car is within 30 meters of the user, we fit the maximal cepstral coefficient computed from the audio stream sampled from the phone to a distance learned via regression. Past 30 meters, the maximal cepstral coefficient levels off to a base value as shown in Figure 7, and we cannot distinguish different distances past this point. This model inherently provides coarse distance classification: greater than 30 meters away and less than 30 meters away. Additionally, if a car is further than 30 meters away from a pedestrian, it is not critical for the user to know the exact distance of the car. As such, PAWS Low-Energy entirely removes the coarse-grained distance classifier and uses only the regression model introduced in Section II-D for both coarse and fine-grained distance estimation.

In the direction classifiers employed in PAWS, relative delay and relative power features are used for classification. Since relative power is no longer available, we can only use relative delay to compute the angle of arrival of the car. According to [21], [22], [23], relative delay is sufficient for computing the direction of arrival of a sound source. One option is to train another eight direction classifier using only relative delay features. If the position of the microphones are known in advance, a second option is to employ classical geometric methods, such as triangulation, to directly compute the location of the source with respect to the user. However, the accuracy of triangulation methods depends heavily on the distance between microphone sensors and the sampling frequency of the audio streams.

Despite the higher localization granularity that can be achieved, a small amount of noise can cause large amounts of direction and distance error in classical geometric methods. In the following section, we present AvPR, a training-based direction of arrival estimation method that provides the same localization granularity as classical triangulation methods, while also being more robust to environmental noise.

C. Angle via Polygonal Regression

AvPR takes advantage of the idea of using data to build a model to learn a physical phenomena and reduce the influence of noise, similar to a machine learning classifier. In the PAWS system, we trained the direction classifier by having someone (or a mannequin) wear the headset and playing a sound source (either generated or a real car sound) in each of the eight directions we classify. Figure 15 displays the relative delay measurements obtained by playing white noise in each of the eight directions. The coordinates of each three-dimensional point corresponds to one of the three relative delay measurements computed in the front-end. We see that samples obtained from all eight directions form a circular shape on a relatively flat plane in three-dimensional space. From this representation, we can easily see that each point within this circular structure maps to a direction with respect to the user. A standard machine learning classifier would disregard this insight and learn boundaries between each of these eight directions to classify future observations into one of these eight categories.

Instead, AvPR provides direction estimates with similar granularity to triangulation methods by creating a mapping between direction, θ, and the circular structure that the relative delays exhibit. For each direction, θ_i, that we take measurements from to build the AvPR model, there is a sample

mean, μ_i, and variance, σ_i^2. As shown in Figure 16, we can approximate the structure of the relative delays, and therefore any angle between two angles for which we have observations for, as an n-sided polygon by interpolating between adjacent direction observation means and variances using linear splines, where n is the number of directions we use to calibrate the model (eight is shown in the example). Now we have interpolated a probability distribution, $x_\theta \sim D(\mu_{x_\theta}, \sigma^2_{x_\theta})$ for all possible directions, $0 \leq \theta \leq 2\pi$, where x is the vector of relative delay observations obtained from the ASIC.

Now that the model is built, whenever a new observation of relative delays, x, arrive, we can estimate the direction, θ^*, by optimizing over the observed and interpolated direction distributions. There are multiple ways to accomplish this, but a common method is to use a maximum likelihood estimator and assume that a car has an equal probability of appearing at any direction, as shown in Equation 5.

$$\theta^* = \max_{0 \leq \theta \leq 2\pi} \mathrm{P}(\mathrm{x}|\theta) \tag{5}$$

This optimization problem is computationally expensive because we have to optimize over an uncountably infinite number of angles. We considered two ways to simplify this optimization. The first method is to quantize the angles into bins and optimize over the finite number of bins. However, this method is very similar to classification: the less bins we divide the angles into, the lower the computation and granularity of our estimator. The second method is to simplify the model by assuming that the variance, σ_θ^2, of all directions θ, are equal. Then, the optimal angle corresponds to the point on the polygon that the observation is closest to, which only involves computing the length of the normal segment from the observation to each side of the polygon, as shown in Figure 17. If the normal segment extends beyond the polygon, then the distance between observation and the polygon side is computed as the distance between the observation and the closest endpoint of the polygon segment. We adopt the second optimization method into AvPR because we preserve the granularity of direction estimation, while also being computationally efficient.

In addition to being more granular than the classification method, AvPR also requires less calibration points for greater granularity than the classifier method employed in PAWS. In theory AvPR has the same granularity as the classical triangulation methods as long as the clusters of relative delay calibration points can form a simple polygon. This is because a simple polygon is two-dimensional and can capture all angles 360 degrees around the user. Since the simple polygon with the least number of vertices is a triangle, AvPR can obtain the same granularity of direction estimates as triangulation methods while only requiring three training directions to build the model. The classification methods require n calibration directions to categorize new observations into n different directions. In Section VI, we compare AvPR with the classifiers employed in PAWS as well as an implementation using triangulation and show the robustness of AvPR over these existing methods. We summarize the steps for building the AvPR model and estimating new directions.

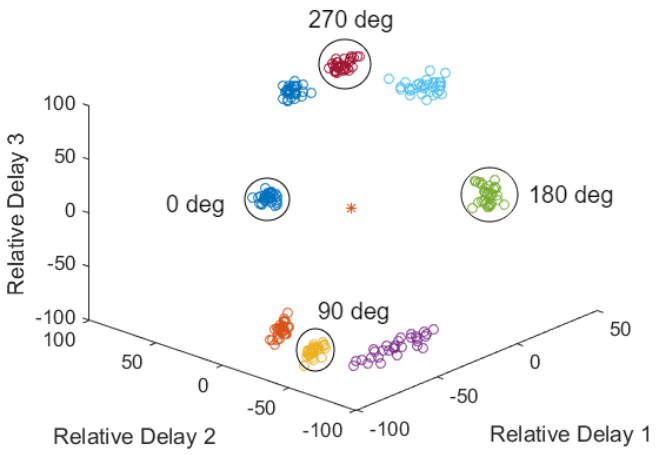


Fig. 15. Plot of relative delay points from playing white noise in eight directions around the user.

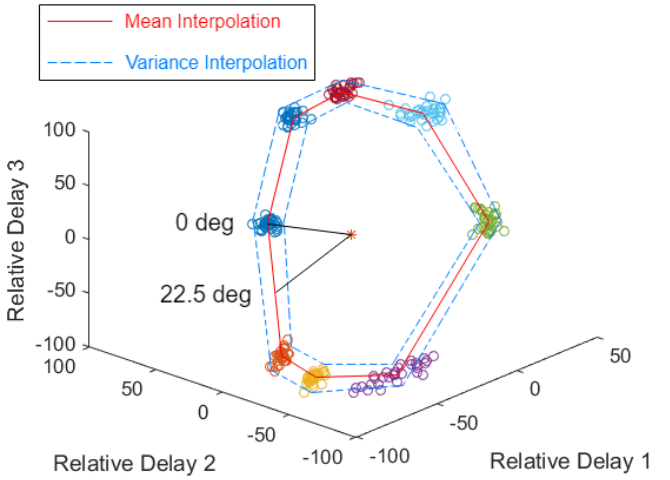


Fig. 16. AvPR mean and variance interpolation between sampled directions to create the polygon model used to estimate all other directions.

1) **AvPR Training Phase**
   a) Play sounds at n directions around headset and record the relative delays as features to obtain calibration directions.
   b) Interpolate between the sample means of adjacent directions. The resulting n-sided polygon in the feature space corresponds to the car's angle of arrival around the user.
2) **Estimating Direction with AvPR**
   a) For a new observation, x, containing the relative delays sampled from the front-end platform, find the point on the trained polygonal model that is closest to the new observation in the feature space of relative delays.
   b) This closest point maps to the direction that AvPR estimates the sound source is coming from.

D. PAWS Low-Energy System Architecture

The PAWS Low-Energy front-end and smartphone block diagram has the same flow as PAWS, but some of the modules within the flow are modified and updated using techniques

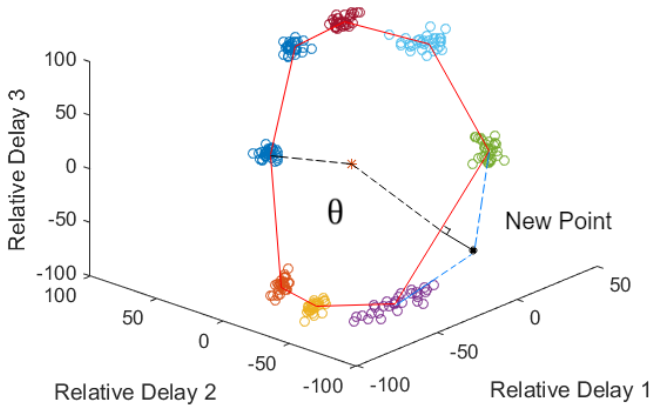


Fig. 17. Estimating a new direction with AvPR. Blue dashed lines are example cases of computing the shortest distance from the new point to a segment. The solid black line denotes the path from the new point to the point on the polygon that is closest to the new point.

introduced in this section. In the front-end platform, PAWS Low-Energy replaces the MCU with the ASIC. This switch occurs in the module enclosed by the upper red box in Figure 8. In the smartphone pipeline, the phone no longer receives relative power and zero-crossing rate features. The lower left red box in Figure 10 marks the area of this difference between PAWS and PAWS Low-Energy. Additionally, the direction and distance classifiers for localization in PAWS are replaced by AvPR and regression of maximal cepstral coefficients respectively. The lower red box in Figure 8 along with the upper and lower right boxes in Figure 10 show where these differences occur in the front-end and smartphone data pipelines. The car and honk detectors remain the same between both PAWS and PAWS Low-Energy.

V. Platform Evaluation

In this section, we first analyze and compare the real-time performance and timing analysis of each component in the PAWS and PAWS Low-Energy systems. Second, we analyze and compare the power consumption of both systems.

A. Real-Time Performance

In this section, we discuss and compare the real-time performance of PAWS and PAWS Low-Energy, the timing constraints involved, and the design decisions involved to meet them. Specifically, we will analyze the timing on the feature extraction in the front-end platform (the headset), the BLE transmission from the front-end to smartphone, and the detection and localization pipeline in the smartphone. Response time is crucial for our system, as a few milliseconds can make a difference in saving the life of a user.

1) Front-end Feature Extraction: The first part of the data flow in PAWS and PAWS Low-Energy is sampling audio and extracting features. In PAWS, the embedded front-end hardware is handling 32kSamples/s with 8 bits per sample for each of the 4 channels with MEMS microphones. To minimize latency, we compute features in 100ms windows every 50ms in a pipeline fashion. This means that features are being calculated every 50ms with 50% window overlap. The MCU uses a dedicated ADC module with direct memory access (DMA) to leave more CPU cycles available for feature calculation. The ADC is continuously sampling audio and storing them in RAM while features from the previous frame are being calculated. The data transfer from the MCU to the BLE module is also done via a dedicated UART module. In order for this pipeline to work in real-time, all features from the current frame must be calculated before the acquisition of the following frame ends, and the UART module must finish sending the current feature vector before the next feature is ready to be sent. The timing of the different parts of this pipeline can be seen in the upper flow of Figure 18. The features calculation consumes 36ms of the available 50ms in one time slot, and the UART module completes each feature vector transmission in 1.9ms.

In PAWS Low-Energy, the MCU is replaced with the ASIC. The ASIC extracts the sign of the audio stream and provides 8-bit relative delay measurements at 50kS/s per channel for three channels. However, we are unable to transmit 50kS/s per channel worth of 8-bit relative delay measurements to the smartphone due to bandwidth limitations of BLE. To keep the same wireless transmission rate of PAWS, we only read one set of relative delays per channel every 2500 samples and transmit to the phone. This yields a transmission rate to the phone of 20 measurements per second. In order for the BLE module to support a transmission rate of 20 measurements per second per channel, the module must be able to read one set of relative delay measurements in less than 50 ms from the ASIC, which uses a custom communication protocol. From our measurements, the BLE module requires 9μs to completely read one relative delay measurement from each channel, which fits our timing requirements for real-time operation. The timing for the ASIC to BLE module extraction for PAWS Low-Energy is shown in the bottom flow of Figure 18.

Since the BLE module is directly reading the relative delays from the ASIC, it does not need to spend time to compute relative delay; it can directly send this value to the smartphone. As such, PAWS Low-Energy directly sends data to the smartphone after only spending 9μs extracting a measurement from the ASIC, while PAWS requires 1.9ms for the BLE module to read the relative delay measurement from the MCU on top of the 36ms of computing the relative delay. This is a significant decrease in latency over PAWS, which will effect the wireless transmission latency test, detailed in the following section.

2) Front-end to Smartphone Wireless Transmission: Another crucial timing aspect of the system is the latency to transmit the features from the BLE module to the smartphone. This latency will not only add to the response time of the system, but it can also cause a mismatch between the vehicle detection and its localization. If the temporal-spatial features calculated in the front-end hardware take too long to reach the smartphone, the location estimation displayed to the user might refer to a different sound source than the vehicle that the system just detected. To verify that the smartphone will receive the data within an acceptable time interval, an adaptation to the system was made, as shown in Figure 19. A button was

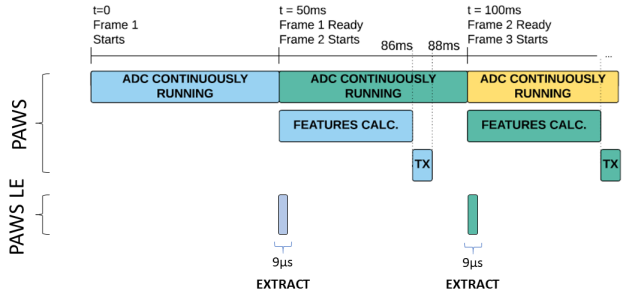


Fig. 18. Pipeline of the PAWS and PAWS Low-Energy feature extraction process. In the PAWS flow, the "Features Calc." block represents all the operations involved in the features extraction, and "TX" represents the UART communication between the MCU and BLE module. In the PAWS Low-Energy flow, "Extract" refers to the BLE module reading relative delays from the ASIC.

simultaneously connected to one of the inputs of the front-end hardware and the microphone input of the smartphone (as the regular microphone buttons). A verification app was developed to compare the difference between the time when the button-press event was detected by the smartphone application and when the smartphone received the data packet containing the same event. All aspects of the systems, including firmware, remain equivalent to the setup for standard operation. Both PAWS and PAWS Low-Energy were tested.

The average delay of the PAWS wireless transmission latency is on the order of 55ms as shown in Figure 20. Since the event can be captured by the MCU anywhere within the 50ms sampling windows, this latency is not expected to be lower than the 38ms required for the calculations and transmission. However, due to randomness in the delay on the smartphone path, a few samples on the histogram have lower latencies. Figure 21 shows that the wireless transmission latency of PAWS Low-Energy is around 27 ms, which is much lower than PAWS. This huge improvement is from the addition of the ASIC, as explained in the previous section. The ASIC updates relative delay measurements at every audio sample, and the BLE module must only read from the ASIC. However in PAWS, the MCU must wait for an entire window (50ms) of samples before spending more than half of a window (36ms) extracting and transmitting features to the BLE module, which adds significant delay to the pipeline. As such, we have shown that PAWS Low-Energy improves upon the front-end feature extraction + wireless transmission latency of PAWS.

3) Smartphone Processing: Figure 22 shows the execution times of various components inside the smartphone application of PAWS and PAWS Low-Energy. The application runs four threads in parallel. Thread 1 is responsible for getting audio data using the single channel microphone for car detection. We have taken 10 frames per window (448ms) for robust feature calculations. Thread 2 is responsible for receiving acoustic features over BLE. Thread 3 runs the car detector, which takes 86ms. In addition to car detection, thread 3 also runs the distance and direction estimators. PAWS requires merely 2 ms to classify distance and direction because these classifiers use precomputed features from the headset. PAWS Low-Energy requires a slightly less, though similar, amount of time to run its localization algorithms. The UI thread (Thread 4) takes 3ms

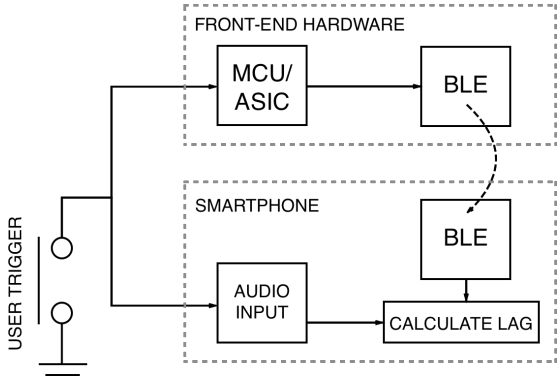


Fig. 19. Block diagram of the test setup for the latency between the features from the front-end hardware and the smartphone.

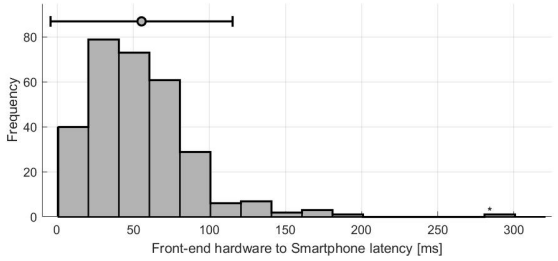


Fig. 20. Histogram of PAWS front-end hardware to smartphone latency acquired with the test setup shown in Figure 19.

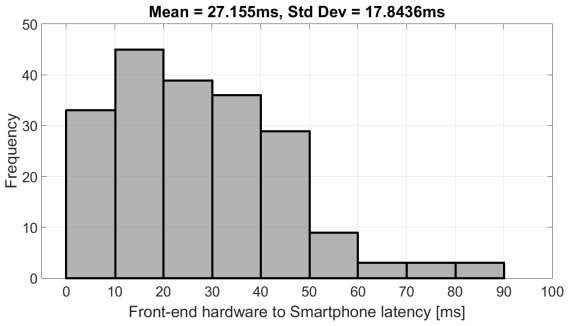


Fig. 21. Histogram of PAWS Low-Energy front-end hardware to smartphone latency acquired with the test setup shown in Figure 19. We see that the mean latency is shorter than PAWS.

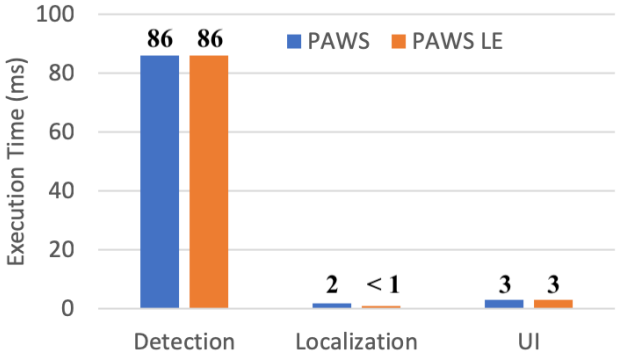


Fig. 22. Execution times of various components of the PAWS and PAWS Low-Energy smartphone app.

to update the UI and to notify the user for both systems as well. The worst case execution time for the PAWS and PAWS

TABLE I
POWER CONSUMPTION AND PRICE BREAKDOWN OF PAWS AND PAWS LOW-ENERGY (**PAWS LOW-ENERGY TOTAL IN BOLD**)

	Idle [mA]	Active [mA]	Unit Price [U$]
MCU (STM32f4)/ASIC	4.37/∼0	50/∼0	3.20
BLE Transciever (nRF52)	0.46	7	6.40
MEMS Mics × 4	0.48 × 4	0.48 × 4	0.40 × 4
Amplifiers × 4	2.34 × 4	2.34 × 4	1.60 × 4
Regulators	0.1/0.6	0.1/0.6	0.50/3.00
Total	16.21/**11.84**	68.4/**18.9**	18.10/**20.60**

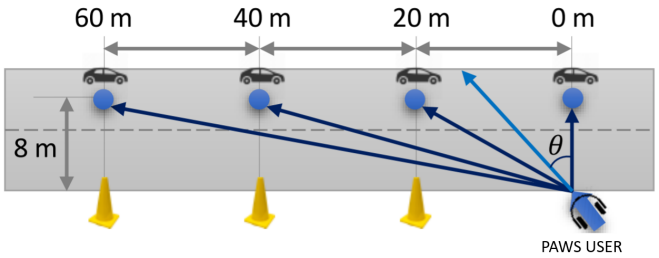


Fig. 23. Experiment scenario in campus street.

Low-Energy apps is 91ms. Because we use a 50% overlap between successive windows for car detection, the PAWSapp runs the full classification pipeline every $448/2 = 224$ms, and detects and localizes cars in 91ms (i.e., in real-time), giving users plenty of time to respond to oncoming dangers.

B. Power Consumption and Price Breakdown

We evaluate the energy consumption of PAWSand PAWS Low-Energy by measuring the power consumptions for both the embedded platform and the smartphone during idle and active states. In the active state, data is processed, features are computed, and results are transmitted to provide danger feedback to the user, whereas in the idle state, the smartphone application is not connected to the headset and most of the clocks in the embedded front-end platform are turned off to conserve power. The sole purpose of the idle state is to conserve power when the user is not using the system (e.g. when the headset is not paired with the phone).

The PAWS embedded platform uses an STM32f4 Cortex-M4 chip as the MCU that samples and extracts features, as well as a BMD-300 module that acts as the BLE transceiver. Operating at 180MHz clock speed, the STM32 MCU consumes the most power at 50mA when active. While not in active use, the power can be reduced to 4.37mA. The Cortex-M4 architecture provides a familiar environment for firmware development with an acceptable energy footprint and a low cost of U$3.20 at major part suppliers. The BMD-300 BLE transceiver module transmitting at 0dBm power consumes 7mA when active and consumes 0.46mA when in idle mode, only transmitting advertisement packets. The BMD-300 module integrates the Nordic nRF52 BLE chipset and antenna in a small footprint component that fits this application for a low price of U$6.40. The other components of the front-end hardware are the 3.3V regulator, the MEMS microphones, and the pre-amplifiers. They consume 0.1mA, 0.48mA, and 2.34mA per component, and cost U$0.50, U$0.40, and U$1.60 per unit respectively. The overall power consumption of the system is below $70mA$, allowing for 17 hours of continuous operation when powered by 3 AAA Alkaline batteries.

The main source of power consumption in PAWS Low-Energy is the BLE module as the ASIC has nW-level consumption. The same MEMS microphones and pre-amplifiers from PAWS were used in PAWS Low-Energy. The ASIC requires multiple voltage inputs to function. As a result, multiple regulators are required, increasing power consumption and price. Since the ASIC requires almost no power to operate, the overall power consumption of the PAWS Low-Energy front-end is more than three times lower than PAWS, allowing for more than two days of continuous operation if powered by standard AAA Alkaline batteries and around half a day of continuous operation if powered by standard CR2032 coin cell batteries. Table I quantifies and compares the power consumption and price breakdown of PAWS and PAWS Low-Energy. The components of PAWS Low-Energy is only around U$2.00 more expensive than PAWS, but result in significant power savings. To obtain the cost estimate for PAWS Low-Energy, we made the assumption that the ASIC would cost the same amount as the STM32f4 MCU if mass produced.

For the smartphone, the most energy consuming component is the display, which is only used to configure the app, and therefore, it is not necessary to keep it always on. The BLE communication consumes about 0.2mA. The energy consumption for the rest of the application is between 0.3uAh to 0.8uAh per frame for both PAWS and PAWS Low-Energy.

VI. REAL-WORLD EVALUATION

To evaluate the end-to-end performance of the complete PAWS and PAWS Low-Energy systems in realistic settings, experiments were conducted in three environments: 1) a street inside a university campus and a residential area, containing pedestrian-borne sounds such as walking and talking; 2) by the side of a highway, with wind being the most prevalent non-car sound; and 3) in a metropolitan area, where both pedestrian-borne and wind sounds are common.

A. Experimental Setup

1. Campus street and residential neighborhood. The first experiment was done in a campus street and a residential neighborhood with a speed limit of 25mph. The background sounds in this environment were mainly pedestrian-borne (e.g. groups of people walking and talking). To evaluate PAWS, we used three fixed markers (yellow cones) on the sidewalk and the PAWS app to evaluate the detection, direction, and distance accuracy. Every time a vehicle passed a cone, a volunteer raised a flag and the event was logged in the PAWS smartphone application. The setup is shown in Figure 23. The experiment was repeated multiple times. Each time the user faced the road at a different angle, θ, so that we could test the accuracy of the direction and distance estimation for as many different angles as possible. For PAWS Low-Energy, we recorded time-stamped video to obtain the ground truth.

2. Side of highway. The second experiment was done by the side of a highway (NC HWY-54) where we observe a constant flow of cars of diverse models, e.g., sedans, SUVs, trucks, and buses. The speed limit for the vehicles in this segment of the

TABLE II
SUMMARY OF DEPLOYMENT EVENTS.

Deployment	User (Facing Angle)	Honks	Car Events
Metro Area	0^o, $\pm 45^o$, 180^o	48	165
Campus	0^o, $\pm 45^o$, 90^o, $\pm 135^o$, 180^o	0	97
Highway	0^o, $\pm 45^o$, 90^o, $\pm 135^o$, 180^o	0	65

highway is 45 mph as it is close to residential areas. In this experiment, we were exposed to less pedestrian-borne noise, but experienced heavy wind noise due to the location of the highway and because we were in hurricane season. For ground truth collection, we marked the road in the similar way as we did in the campus street. However, unlike the campus street, cars on the highway were driven at a higher speed (around 50-55 mph) and they were large in number. Therefore, instead of appointing human volunteers, we recorded a time-stamped video and analyzed the video offline to obtain the ground truth.

3. Metropolitan area. The third experiment was done in the streets of Manhattan, New York, where the number of cars, adjacent streets, and buildings in the surrounding area is very dense. This environment is replete with sounds commonly found in the first two scenarios, including wind and the bustle of pedestrians and wildlife (e.g. birds and pets). Just as with the second experiment conducted near a highway, a time-stamped video was recorded and analyzed offline to obtain ground truth. To evaluate the detection, direction, and distance classifiers of PAWS and PAWS Low-Energy, the estimator outputs were logged and compared against the ground truth. During all the experiments we simulated a distracted pedestrian's ability to detect cars by logging car events while listening to music in parallel with PAWS.

Table II provides statistics of the deployment in all environments. The table shows how the PAWS and PAWS Low-Energy user faced the road, and the number of logged honks and car events. Though the experiments presented in this section were conducted with a stationary user, we observe similar performance when the user moves at walking pace. We did not run experiments in scenarios where the user is moving at higher speeds, but will explore this avenue in future work.

B. Results

1) Car Detection: We measure the car detection accuracy of PAWS and compare its performance with that of the ground truth collector's and distracted user's reports. Since PAWS Low-Energy uses the same car detector, the results of PAWS Low-Energy are aggregated with the results of PAWS. Figure 24 compares the exact counts of total logged approaching car events for all environments. We see that almost all the cars logged by the ground truth collector have been identified by PAWS, whereas the distracted participant missed about 19%-36% of them. This shows that PAWS is a highly efficient system for detecting and alerting pedestrians of approaching cars. In summary, the car event detection accuracy is 97.30%, 99.48% and 95.59% in metro area, campus and highway respectively. Additionally, a confusion matrix for the detection classifier running on PAWS is presented in Figure 25 for the metro area. The difference in the counts shown in Figure 25

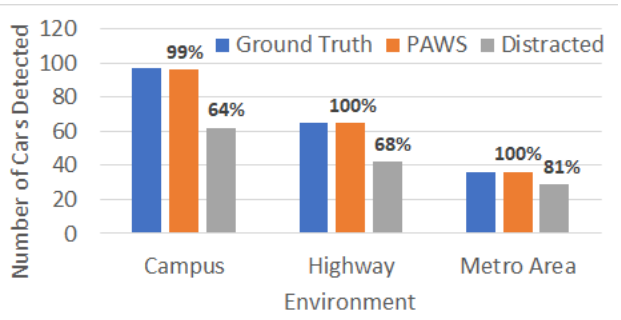


Fig. 24. Car detection performance.

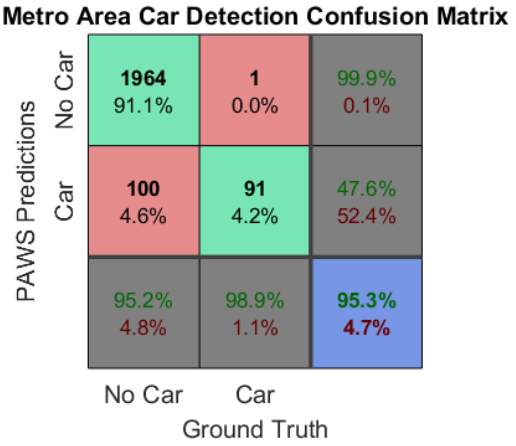


Fig. 25. Metro area car detection confusion matrix.

and Figure 24 is that in Figure 24, the values correspond to car events. For instance if one car passes by the user, it will count as a single car event in this figure, but PAWS will have multiple frames or windows it processes that will show up as car detections from the raw classifier outputs. Figure 25 on the other hand displays the confusion matrix for each individual frame computed by the PAWS application. We see that only one frame was misclassified as a noncar in the case where a car was present, and around 5% of the noncar samples were misclassified as cars. These values show that PAWS has fairly low false positive and false negative rates as well as high true positive and true negative rates. In other words, PAWS is able to correctly detect the presence of cars and reject cases where no car is present in common urban environments, rich with wind and pedestrian-borne noises in the background.

2) Localization Performance: In this section we will present evaluation on the direction and distance estimators of PAWS and PAWS Low-Energy. First, we compare the performance of the direction estimators of PAWS, the AvPR method employed in PAWS Low-Energy, and the classical triangulation method. The results of the PAWS direction classification is shown in Figure 28 for all environments and the eight directions that we trained the classifier to discern. We assume that the accuracy of the directions reported by the ground truth collector is accurate. Each reported direction from the ground truth collector is mapped to the classification results of PAWS. We observe that the average accuracy of the direction classifier over all directions is 86.7%.

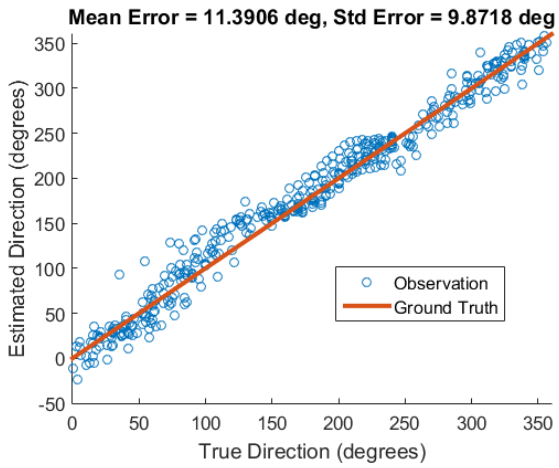


Fig. 26. Estimated direction vs. true direction estimated via AvPR.

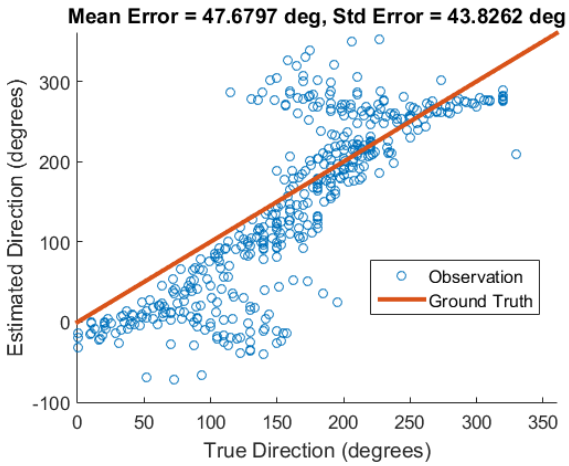


Fig. 27. Estimated direction vs. true direction estimated via triangulation.

Figure 26 plots the mean and variance of the estimated vs. true angle of the AvPR method employed by PAWS Low-Energy. We see that the average error over all directions is around 11 degrees, which is less than the 12.5 degree sections that the classifier in PAWS outputs. This shows that AvPR is more accurate and granular than the classification approach employed by PAWS. We also show the mean and variance of the estimated vs. true angle of the car computed via triangulation in Figure 27 as a comparison. The average error is around 45 degrees, which is larger than the error of AvPR. The standard deviation of error for AvPR is around 10 degrees compared to 44 degrees for triangulation, which shows that AvPR provides more consistent predictions than triangulation. The mean and variation in errors show that AvPR outperforms AvPR for direction estimation.

Next, we present the results of the distance estimators of PAWS and PAWS Low-Energy. Recall that PAWS performs coarse distance classification and only performs fine-grained distance estimation if the car is detected to be within 30 meters, while PAWS Low-Energy leverages the coarse-grained estimator found inherently within the fine-grained estimator for coarse-grained and fine-grained estimations. Figure 30

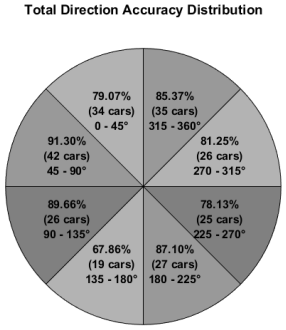


Fig. 28. Combined accuracy for PAWS directions prediction.

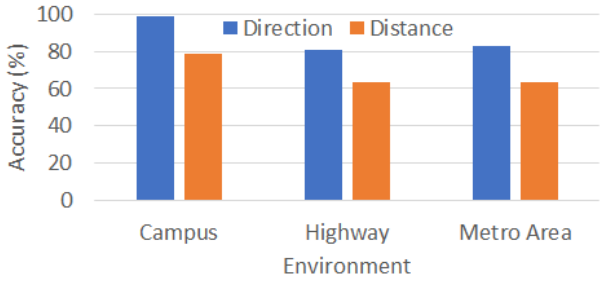


Fig. 29. Car localization accuracy.

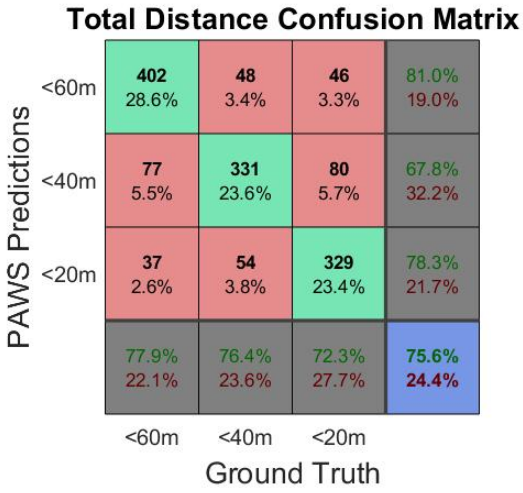


Fig. 30. confusion matrices for distance estimation.

shows the combined confusion matrix for all distance predictions of the coarse-grained classifier. Each distance section presents an accuracy of 77.9%, 76.4%, and 72.3% for ranges from above 60m to 40m, to 20m from the user. The overall accuracy the coarse-grained estimator is 75.6%. Figure 31 plots the estimated distance vs. true distance of cars driving towards the user wearing our systems. We see that for cars within 30 meters, the distance estimator has an average error of 2.8 meters.

Figure 29 summarizes the direction and distance classifier results for all environments. We observe that the overall accuracy of the distance classifier is $63\% - 78\%$, and that the average direction classifier ranges from $80\% - 98.5\%$ depending on the environment.

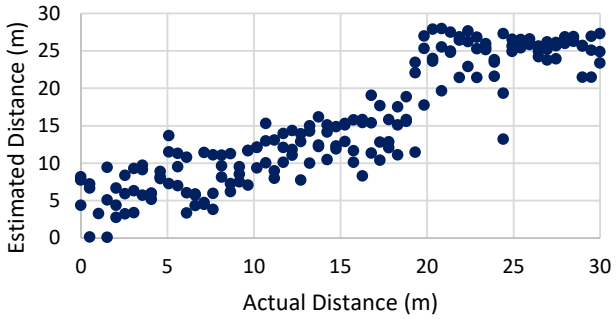


Fig. 31. Estimated distances for cars within 30m are on average 2.8m off of actual distances.

C. Limitations and Future Work

In this work, we present novel, audio-based, wearable methods for enhancing pedestrian safety that is effective in typical urban scenarios, providing accurate and real-time alerts of vehicle presence and location. However, we recognize that our system is still at the research level and not ready for commercialization, as there are several scenarios in which PAWS and PAWS Low-Energy are ineffective. We discuss these situations next.

1) Noisy Streets: PAWS is designed to detect the presence of cars in real-world environments. Streets may contain diverse kinds of noise, some of which may be different from the ones we have calibrated our system for. PAWS should be trained in as many scenarios as possible for robustness.

Additionally, just as if a camera in a vision-based approach is unable to see the car (e.g. car is not in line of site or if it is dark outside), if for any reason the microphones are unable to pick up the sounds of the car (e.g. microphones are covered or noise overpowers everything in the environment), then PAWS would be rendered ineffective. We are currently looking into other sensing modalities, vehicular networking methods, and crowdsourcing to overcome the limitations in our audio processing methods.

2) Nearby Cars: The current design of PAWS considers only the positions of vehicles relative to the user, but not their trajectories. We can foresee occasions where a pedestrian is walking parallel to a busy road, and the system is giving warnings, even though the user is not in danger of being hit. Future work that takes into account the trajectory of both the vehicle and the user is under development.

3) Multiple Approaching Cars: The presence of multiple cars approaching the user can impair the reliability of the system. It does not matter if there are multiple cars present or if there is only a single car, PAWS will reliably detect the presence of vehicles, using the methods presented in Section III-C1, at no additional computational cost if multiple cars are present. The localization portion of PAWS and PAWS Low-Energy both use relative delay computed via time-domain cross-correlation methods, which are dominated by the highest energy source (e.g. the loudest vehicle). In summary, PAWS can detect the presence of vehicles and localize the loudest vehicle. PAWS cannot estimate the number of vehicles present, nor localize multiple present vehicles. We are currently investigating sound source source separation and multiple sound source localization techniques to overcome this challenge.

It should be noted that although PAWS is unable to localize multiple oncoming cars, the majority of accidents occur when there are less cars in the streets. According to the National Highway Traffic Safety Administration (NHTSA), 70% to 80% of all pedestrian-related accidents in the United States occur between 6PM to 6AM [24], when there are arguably less cars on the road than during the day. Additionally, the NHTSA also reports that 90% of accidents that included the death of a pedestrian involved a single vehicle. These statistics suggest that despite its shortcomings in handling multiple car cases, PAWS can still handle the more common scenarios involving accidents with fewer vehicles.

VII. Related Work

Object recognition and localization have been vastly explored in the literature. Almost all of them mirror techniques that are present in nature, such as the use of stereo imaging [25], ultrasonic radars [26], and acoustic source localization [27]. In vehicular tracking, video based approaches have been widely used [28][25][29]. The amount of information that can be extracted from images is undoubtedly greater than any other types of sensors. Commonality in vehicles' shapes and standardized road signs have enabled machine learning algorithms to identify and predict the movement of cars [30]. Although such systems offer outstanding solutions for devices that can be hosted in large platforms, e.g. in an autonomous car for collision prevention [31], these are not suitable for use in wearable systems. A major limitation is the high computational requirements for real-time image processing. Another major issue is the privacy of the user. Video can reveal an alarming amount of personal identifiable information.

Active techniques like radar and LIDAR can certainly be used to detect the presence of obstacles and even some of its spatial behaviors [32][33], but such solutions face great challenges in classifying what those obstacles are. This is particularly problematic in urban environments where moving and stationary obstacles are abundant, but only a few are real threats to the user. On the implementation side, the inherently high power dissipation of active transducers are usually discouraging for portable devices.

Passive audio sensors, on the other hand, provide enough information to allow classification and localization of the source with less computational and power requirements. But, unlike other techniques already published [34][27][23], PAWS uses machine learning algorithms to improve its predictions. By doing so, the system requires a large amount of learning data, but gains in flexibility, speed, and complexity. Audio classification has been used for event detection like, coughing detection [35], gun shot detection [36], human activity (e.g. talking, crying, running etc) detection [37]. These works mostly focus on identifying prominent sounds like gun shots or shouting rather than noise-like car sounds. [38] classified subtle sounds like keyboard typing, door knock etc. but all of the sounds were in an isolated environment not in real life noisy environment. [39] considered bus and trucks as events but had a very low accuracy of 24%. Other signals like video [40] [41] or seismic signals [42] have been used for vehicle detection but are not suitable for a wearable system like PAWS.

Other works that leverage sensors to enhance pedestrian safety utilize sensors placed on the shoe [43] or the camera on the smartphone [44], but these approaches are either unable to detect and localize cars or provide limited coverage.

In recent years, developments in vehicle to vehicle, pedestrian, and infrastructure communications are beginning to allow cars and pedestrians to directly communicate with each other in close proximity. Dedicated Short Range Communication (DSRC), a wireless protocol developed specifically for vehicular networking, is becoming a popular protocol in vehicular networks to transmit information and alerts [45][46], but is not natively supported on smartphones and require modifications to existing vehicles. Solutions that leverage standard WiFi or cellular protocols generally require every car to have a wireless transmitter [47][48], do not meet timing and latency requirements [49][50], or modify the SSID of smartphone WiFi beacons [51] and cannot be implemented on a smartphone with standard privileges. Additionally, a single pedestrian only requires our custom headset to receive the full functionality of PAWS and PAWS Low-Energy. In contrast, V2X systems commonly require modifications to support specific wireless protocols on all passing cars before a single pedestrian can benefit.

VIII. Conclusion

This paper presents PAWS and its low-power variant PAWS Low-Energy, a wearable system that uses multiple audio sensors to protect pedestrians by identifying and localizing approaching vehicles. PAWS is carefully designed to recognize honks and noises of an approaching vehicle. Using machine learning algorithms and signal processing techniques, PAWS is able to identify honks and tire/engine sounds with near 100% precision across all tested environments. It further provides feedback on the direction of the sound source with $80\% - 98.5\%$ accuracy and predicts the distance from the user with $62\% - 78\%$ accuracy. As technology evolves and new dangers surround modern cities, innovative safety solutions must arise to uphold the welfare of common citizens.

IX. Acknowledgements

This research was partially supported by the National Science Foundation under Grant Numbers CNS-1704899 and CNS-1815274. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Columbia University, NSF, or the U.S. Government or any of its agencies.

References

[1] D. de Godoy, B. Islam, S. Xia, M. T. Islam, R. Chandrasekaran, Y. Chen, S. Nirjon, P. R. Kinget, and X. Jiang, "Paws: A wearable acoustic system for pedestrian safety," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, April 2018, pp. 237–248.

[2] K. Shaver, "Safety experts to pedestrians: Put the smartphones down and pay attention," September 2014, [Online]. [Online]. Available: https://www.washingtonpost.com/local/trafficandcommuting/safety-experts-to-pedestrians-put-the-smartphones-down-and-pay-attention/2014/09/19/278352d0-3f3a-11e4-9587-5dafd96295f0_story.html

[3] M. Park, "Injuries while walking with headphones tripled, study finds," January 2012, [Online]. [Online]. Available: http://thechart.blogs.cnn.com/2012/01/16/injuries-while-walking-with-headphones-triple-study-finds/

[4] S. Li, X. Fan, Y. Zhang, W. Trappe, J. Lindqvist, and R. E. Howard, "Auto++: Detecting cars using embedded microphones in real-time," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 70, 2017.

[5] K. G. Shin and Y.-C. Tung, "Real-time warning for distracted pedestrians with smartphones," Sep. 25 2015, uS Patent App. 14/865,262.

[6] M. Galleso, *AirPods: An Easy Guide to the Best Features*. CreateSpace Independent Publishing Platform, 2016.

[7] A. Champy, "Google pixel budswireless headphones that help you do more," October 2017, [Online]. [Online]. Available: https://www.blog.google/products/pixel/pixel-buds/

[8] C. J. Plack, *The sense of hearing.* Lawrence Erlbaum Associates Publishers, 2005.

[9] R. S. S. Molau, M. Pitz and H. Ney, "Computing mel-frequency cepstral coefficients on the power spectrum," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*. IEEE, 2001.

[10] A. Martin, D. Charlet, and L. Mauuary, "Robust speech/non-speech detection using lda applied to mfcc," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 1. IEEE, 2001.

[11] T. Kinnunen, E. Chernenko, M. Tuononen, P. Fränti, and H. Li, "Voice activity detection using mfcc features and support vector machine," in *Int. Conf. on Speech and Computer (SPECOM07)*, 2007.

[12] J. Portelo, M. Bugalho, I. Trancoso, J. Neto, A. Abad, and A. Serralheiro, "Non-speech audio event detection," in *Acoustics, Speech and Signal Processing, ICASSP 2009.* IEEE, 2009.

[13] S. G. Koolagudi and K. S. Rao, "Emotion recognition from speech: a review," *International journal of speech technology*, vol. 15, no. 2, 2012.

[14] N. Bhave and P. Rao, "Vehicle engine sound analysis applied to traffic congestion estimation," in *Proc. of International Symposium on CMMR and FRSM2011*, 2011.

[15] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *Automatic Control, IEEE Transactions on*, vol. 26, no. 1, 1981.

[16] B. Van Den Broeck, A. Bertrand, P. Karsmakers, B. Vanrumste, M. Moonen *et al.*, "Time-domain generalized cross correlation phase transform sound source localization for small microphone arrays," in *Education and Research Conference, 2012 5th European DSP*. IEEE, 2012.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, 2001.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, 1995.

[19] D. de Godoy, X. Jiang, and P. R. Kinget, "A 78.2 nw 3-channel time-delay-to-digital converter using polarity coincidence for audio-based object localization," in *IEEE Custom Integrated Circuits Conference*, 2018.

[20] S. Wolff, J. Thomas, and T. Williams, "The polarity-coincidence correlator: A nonparametric detection device," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 5–9, January 1962.

[21] M. Omologo and P. Svaizer, "Acoustic event localization using a crosspower-spectrum phase based technique," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, vol. 2. IEEE, 1994, pp. II–273.

[22] J. C. Chen, K. Yao, and R. E. Hudson, "Source localization and beamforming," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 30–39, 2002.

[23] J.-M. Valin, F. Michaud, and J. Rouat, "Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering," *Robotics and Autonomous Systems*, vol. 55, no. 3, 2007.

[24] "2016 pedestrians traffic safety fact sheet," March 2018. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812493

[25] M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision-based vehicle detection," in *IEEE Intelligent Vehicles Symposium*, 2000.

[26] B. Barshan and R. Kuc, "A bat-like sonar system for obstacle localization," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 4, 1992.

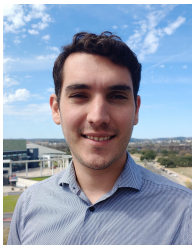
[27] J. H. DiBiase, H. F. Silverman, and M. S. Brandstein, "Robust localization in reverberant rooms," in *Microphone Arrays*. Springer, 2001.

[28] W. Wang, "Reach on sobel operator for vehicle recognition," in *Artificial Intelligence, International Joint Conference on*. IEEE, 2009.

[29] G. J. McDonald, J. S. Ellis, R. W. Penney, and R. W. Price, "Real-time vehicle identification performance using fpga correlator hardware,"

Intelligent Transportation Systems, IEEE Transactions on, vol. 13, no. 4, 2012.
[30] S. Zhou, J. Gong, G. Xiong, H. Chen, and K. Iagnemma, "Road detection using support vector machine based on online learning and evaluation," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010.
[31] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2006.
[32] S.-L. Jeng, W.-H. Chieng, and H.-P. Lu, "Estimating speed using a side-looking single-radar vehicle detector," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 15, no. 2, 2014.
[33] Z. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Synthetic 2d lidar for precise vehicle localization in 3d urban environment," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013.
[34] M. S. Brandstein, J. E. Adcock, and H. F. Silverman, "Microphone-array localization error estimation with application to sensor placement," *The Journal of the Acoustical Society of America*, vol. 99, no. 6, 1996.
[35] A. Harma, M. F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *Multimedia and Expo, 2005. ICME 2005.* IEEE, 2005.
[36] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005.
[37] P. K. Atrey, N. C. Maddage, and M. S. Kankanhalli, "Audio based event detection for multimedia surveillance," in *Acoustics, Speech and Signal Processing, 2006. Proceedings. International Conference on*. IEEE, 2006.
[38] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo, "Clear evaluation of acoustic event detection and classification systems," in *International Evaluation Workshop on Classification of Events, Activities and Relationships*. Springer, 2006.
[39] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference, 2010 18th European*. IEEE, 2010.
[40] D. A. Sadlier and N. E. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Transactions on Circuits and Systems for Video Technology*, 2005.
[41] M. Xu, N. C. Maddage, C. Xu, M. Kankanhalli, and Q. Tian, "Creating audio keywords for event detection in soccer video," in *Multimedia and Expo, 2003.. Proceedings. International Conference on*. IEEE, 2003.
[42] N. Evans, "Automated vehicle detection and classification using acoustic and seismic signals," Ph.D. dissertation, University of York, 2010.
[43] S. Jain, C. Borgiattino, Y. Ren, M. Gruteser, Y. Chen, and C. F. Chiasserini, "Lookup: Enabling pedestrian safety services via shoe sensing," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15. ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2742647.2742669
[44] T. Wang, G. Cardone, A. Corradi, L. Torresani, and A. T. Campbell, "Walksafe: A pedestrian safety app for mobile phone users who walk and talk while crossing roads," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems; Applications*, ser. HotMobile '12. ACM, 2012. [Online]. Available: http://doi.acm.org/10.1145/2162081.2162089
[45] X. Wu, R. Miucic, S. Yang, S. Al-Stouhi, J. Misener, S. Bai, and W. Chan, "Cars talk to phones: A dsrc based vehicle-pedestrian safety system," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, Sep. 2014, pp. 1–7.
[46] Z. Liu, L. Pu, Z. Meng, X. Yang, K. Zhu, and L. Zhang, "Pofs: A novel pedestrian-oriented forewarning system for vulnerable pedestrian safety," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*, Oct 2015, pp. 100–105.
[47] P. Ho and J. Chen, "Wisafe: Wi-fi pedestrian collision avoidance system," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 4564–4578, June 2017.
[48] J. J. Anaya, P. Merdrignac, O. Shagdar, F. Nashashibi, and J. E. Naranjo, "Vehicle to pedestrian communications for protection of vulnerable road users," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 1037–1042.
[49] C. Lin, Y. Chen, J. Chen, W. Shih, and W. Chen, "psafety: A collision prevention system for pedestrians using smartphone," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.
[50] M. Won, A. Shrestha, and Y. Eun, "Enabling wifi p2p-based pedestrian safety app," 2018.
[51] K. Dhondge, S. Song, B. Choi, and H. Park, "Wifihonk: Smartphone-based beacon stuffed wifi car2x-communication system for vulnerable road user safety," in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, May 2014, pp. 1–5.

Stephen Xia received his B.S. in electrical engineering from Rice University, USA, in 2016, and his M.S. in electrical engineering from Columbia University in 2018. He is currently a Ph.D. candidate in the Department of Electrical Engineering at Columbia University, where his current research focuses on designing intelligent systems for mobile/wearable computing, connected health, and the Internet of Things.



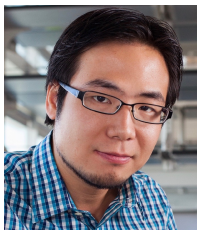
Daniel de Godoy Peixoto was born in Recife, Brazil, in 1988. He received the B.Eng. degree in electrical engineering from the Federal University of Pernambuco (UFPE), Brazil, in 2012, and the M.S. degree in electrical engineering from Columbia University, New York, in 2015, where he received the Columbia Electrical Engineering Department Research Award. He is currently working towards his Ph.D. at Columbia University, where he is focusing on ultra-low-power analog feature extraction front-end integrated circuits for machine-learning audio-based systems. During his studies at Columbia, Daniel was awarded the Science Without Borders Fellowship, from CAPES, Brazil, and the Lemann Foundation Fellowship. His research interest includes ultra-low-power analog and mixed-signal sensor interfaces and their roles in embedded IoT systems.

Bashima Islam received her B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET). From 2016 she has been a Ph.D. student at University of North Carolina at Chapel Hill, USA. Her research interest falls into low power computing and machine learning in resource-constrained devices. Currently, she is working on energy-harvested systems for efficient computing.

Md Tamzeed Islam received his B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET). He is now a Ph.D. student at University of North Carolina at Chapel Hill, USA. He is in his third year of his Ph.D. program. His research interest is in acoustic signal processing and applied machine learning.

Shahriar Nirjon is an assistant professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. Shahriar (who goes by "Nirjon") is interested in Embedded Intelligence—the general idea of which is to make resource-constrained embedded systems capable of sensing, learning, adapting, and evolving in real-time. Research challenges that he deals with include on-device machine learning, RF sensing, real-time issues, and a variety of optimization problems on resource-constrained embedded platforms. His work has applications in the area of smart cities, remote health and wellness monitoring, and the Internet of Things. Nirjon received his Ph.D. from the University of Virginia, Charlottesville in 2014. He has won a number of awards, including two Best Paper Awards at the Mobile Systems, Applications, and Services (MOBISYS 2014), and the Real-Time and Embedded Technology and Applications Symposium (RTAS 2012). Nirjon has worked as a Research Scientist in the Networking and Mobility Lab at the Hewlett-Packard Labs in Palo Alto, CA (2014-2015), and as a Research Intern at Microsoft Research, Redmond, WA (Summer 2013) and at Deutsche Telekom Lab, Los Altos, CA (Summer 2010). Several of his work has been highlighted in the electronic and print media, including the Economist, the New Scientist, and the BBC.

Peter R. Kinget (M'90–SM'02–F'11) received the engineering degree in electrical and mechanical engineering and the Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Belgium, in 1990 and 1996, respectively. From 1996 to 1999, he was with the Bell Laboratories, Lucent Technologies, in Murray Hill, NJ, USA as a Member of Technical Staff with the Design Principles Department. From 1999 to 2002, he held various technical and management positions in IC design and development with Broadcom, CeLight, and MultiLink. In 2002, he joined Columbia University, NY, USA where he currently is the Dept. Chair and the Bernard J. Lechner Professor of Electrical Engineering. From 2010 to 2011, he was with the Université Catholique de Louvain, Belgium, on sabbatical leave. He also serves as an expert on patent litigation and a technical consultant to industry. His research interests are in analog, RF and power integrated circuits and the applications they enable in communications, sensing, and power management. He has widely published in circuits and systems journals and conferences, has co-authored 3 books and holds 32 US patents with several applications under review.



Xiaofan (Fred) Jiang is an Assistant Professor of Electrical Engineering and Computer Engineering at Columbia University and co-Chair of Smart Cities Center at the Data Science Institute. Jiang received his B.Sc., M.Sc., and Ph.D. in Electrical Engineering and Computer Science from UC Berkeley, in 2004, 2007, and 2010, respectively. He was Director of Analytics and IoT Research at Intel Labs China prior to joining Columbia University in 2015. His research interest lies at the intersection of systems and data, with a focus on intelligent embedded systems and their applications in mobile and wearable computing, intelligent built environments, Internet of Things, and connected health. Jiang led one of the earliest projects on IP-based smart-buildings, bringing about the first IPv6/6LowPAN smart metering network and fine-grain real-time energy analytics. His ACme building energy platform was widely adopted by industry and academia. His city-scale air-quality project was featured on China Central Television and Peoples Daily and was successfully incubated into a startup. Jiang has 8 US patents. He has published in top-tier venues with over 3,500 total citations and was awarded Best Paper at IEEE/ACM IPSN '05, Best Demo at ACM SenSys '11, Best Poster at ACM BuildSys '16, Best Paper-Runner Up at ACM BuildSys '17, and Best Demo at ACM IoTDI '18. He has chaired BuildSys '14, IoT Expo '16, ICCCN HoT '17, and is serving as General co-Chair of ACM SenSys '19.