PAWS: A Wearable Acoustic System for Pedestrian Safety

Daniel de Godoy*, Bashima Islam†, Stephen Xia*, Md Tamzeed Islam†, Rishikanth Chandrasekaran*
Yen-Chun Chen†, Shahriar Nirjon†, Peter Kinget* and Xiaofan Jiang*
*Columbia University, †University of North Carolina at Chapel Hill
{dd2697,sx2194,rc3022,peter.kinget,xiaofan.jiang}@columbia.edu, {bashima, tamzeed, yenchun, nirjon}@cs.unc.edu

***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 and implementation consisting of headset-mounted audio sensors, a front-end hardware 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 distance, near 100% precision on the vehicle detection and alert the user with low latency.**

I. Introduction

Smartphones have transformed our lifestyles dramatically, mostly for the better. Unfortunately, listening to music while walking has also 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 making themselves more vulnerable by tuning out the traffic around them [30], 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 [25]. 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.

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.

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 time difference of arrival (TDoA) or other localization techniques. However, these approaches require the user to hold constantly hold their phones steady and to not block the built-in microphone while walking [20] [31], in addition, 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 from him. This problem is further compounded by high levels of mixed noise, typical of realistic street conditions in metropolitan areas.

To tackle these challenges, we develop a segmented architecture and data processing pipeline that partitions computation into processing modules across a front-end hardware platform and a smartphone. Four channels of audio are collected by a microcontroller-based front-end platform from four MEMS microphones that are strategically positioned on a headset. Temporal-spatial features such as relative delay, relative power, and zero-crossing rate are computed inside the front-end platform using the 4 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. We evaluate PAWS using both controlled experiments inside parking lots as well as real-world deployments on urban streets.

We make the following contributions in this paper:

- We create an end-to-end, low-cost, wearable system and smartphone application that provide real-time alerts to pedestrians in noisy urban environments with good accuracies. We demonstrate that inattentive pedestrians can immediately benefit from our system.

This material is based upon work supported by the National Science Foundation under Grant No. 1704899, 1704469, and 1704914.

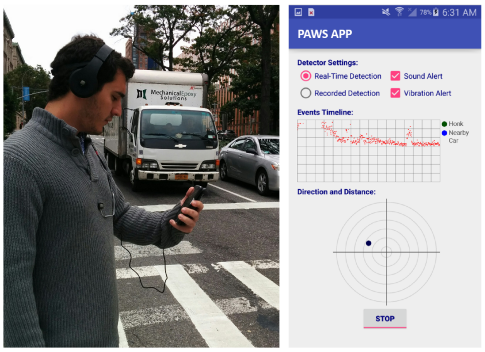


Fig. 1: An inattentive pedestrian wearing a PAWS headset, and a screen shot of the PAWS application user interface.

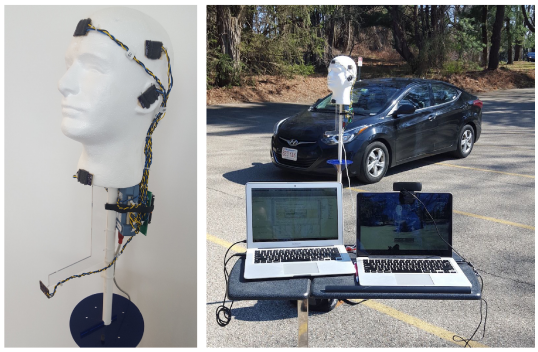


Fig. 2: Validation system: reference mannequin with eight MEMS microphones and data acquisition board in a low-noise controlled experiment setup.

- 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.
- We propose a new acoustic feature 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, with respect to the user, in real-time.
- We share with the community our entire data set, which includes high-fidelity multi-channel audio recordings of moving car sounds, honks, and street noises, that we have collected in a metropolitan area and in a college town.

As the industry is investing heavily in intelligent headphones [8], [14], our hardware-software co-design approach presents a compelling solution towards protecting distracted pedestrians.

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 for 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 phenomenons of the final setup, such as the acoustic shadow of the human head [26] and the approximate spacings 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 recorded audio from a total of 47 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.

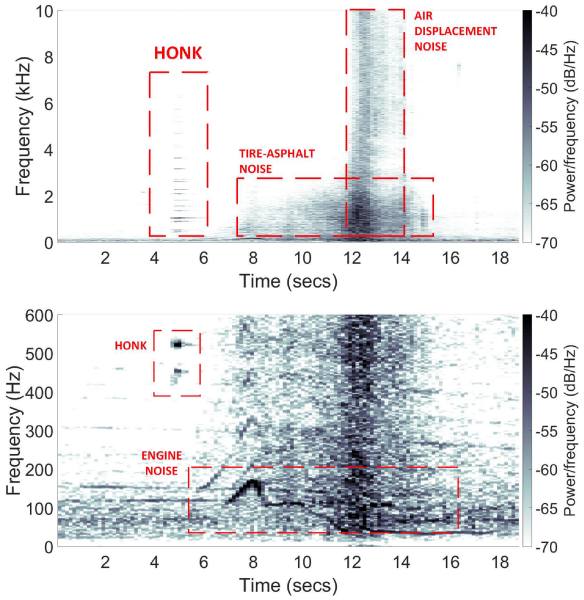


Fig. 3: Spectrogram of one of the recordings from the controlled environment. The car was approaching the mannequin at 25mph.

A. Recording Specifications

In order to characterize the sounds we are interested in, such as an approaching vehicle's tire friction, engine noise, and honks, we conducted controlled experiments in two parking spaces (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 zoomed, 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

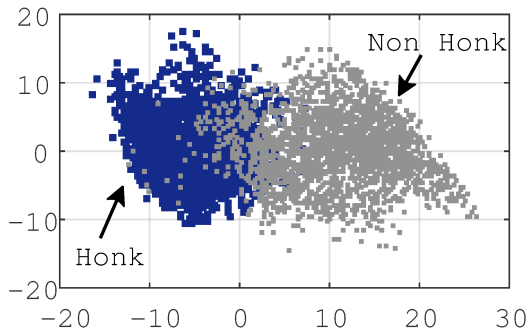


Fig. 4: Distribution of honks and other types of sounds in a 2D feature space.

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 approaching cars at similar speed (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 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.

Honks are louder and, thus, easier to detect than car tire or engine sounds. We analyze the Mel-Frequency Cepstral Coefficients (MFCC) [28] 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 [21] [18] [27] [19] including car sounds [5]. For visualization purpose, we reduce the 13-dimension MFCC features to two dimensions (using PCA [24]) 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 as tire-road frictions. 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 of audio signals. Section III-C1 describes this new acoustic feature.

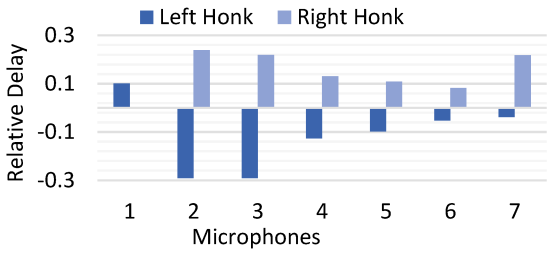


Fig. 5: Normalized relative delays of the microphones for left and right honks.

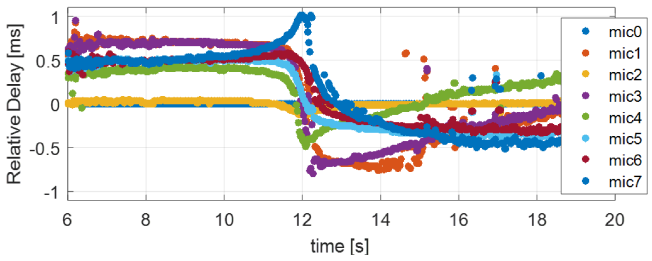


Fig. 6: Relative delay versus time of a car driving past a mannequin from its left to right.

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 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.

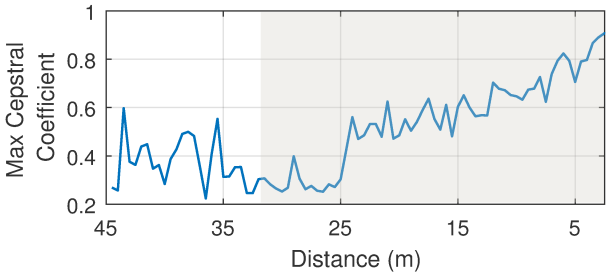


Fig. 7: The maximum cepstral coefficient follows a trend when an approaching car is within about 30m from an observer.

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 by signal-energy and zero-crossing rates, and can be classified accurately using a machine learning classifier.

Therefore, PAWS uses a two-level approach for distance estimation. The first level employs a classifier to 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. Figure 8 shows a scatter plot of actual distances and estimated distances of cars coming toward the user wearing the PAWS headset during a controlled experiment.

III. Overview of PAWS

PAWS is a wearable headset platform together with a 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 9. Four of the MEMS microphones, labeled MIC1 to MIC4, are distributed over the user, at the left and right ear,

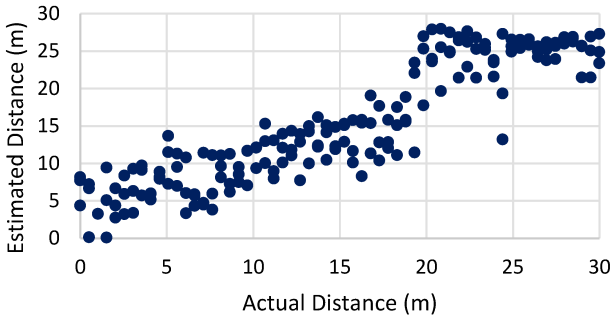


Fig. 8: Estimated distances for cars within 30m are on average 2.8m off of actual distances.

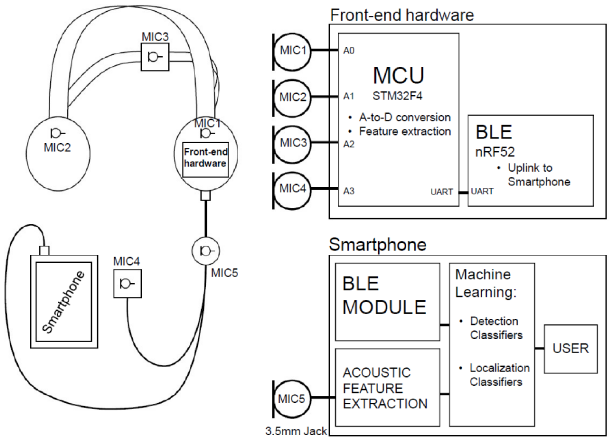


Fig. 9: A block diagram of PAWS.

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 an application running on the smartphone. 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 the large volume of raw audio data

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. 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 to reduce energy 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.

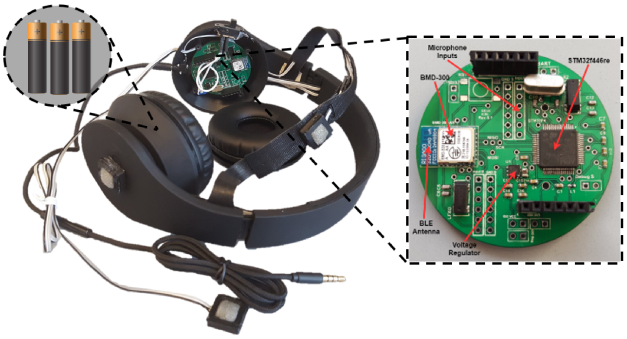


Fig. 10: (Left) Teardown of the PAWS headset; the front-end hardware is exposed inside the left ear housing. (Right) Close up of the PCB that comprises the PAWS front-end hardware.

In order to demonstrate PAWS's system architecture and algorithms, off-the-shelf components were used to build the system. We are in the process of building a nano-watt custom IC to further reduce energy in future iterations. As shown in Figure 9, 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 10.

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 IV.

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 channel 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 [35], these limits on the interest interval of the cross correlation result make the time-domain calculation of the cross correlation more efficient than frequency domain approaches.

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 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 and 10-element acoustic features over BLE in 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 11 shows the data processing pipeline of the PAWS smartphone application. The application implements a two-stage pipeline for detecting and localizing cars, respectively.

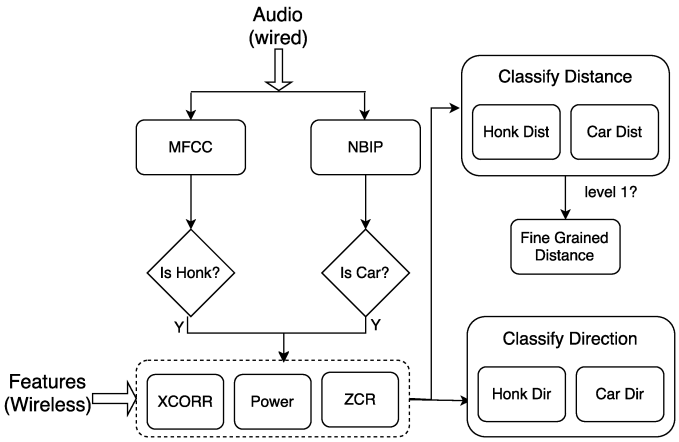


Fig. 11: Smartphone data processing.

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, termed as the *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 car noises. The steps to compute the NBIP features are as follows.

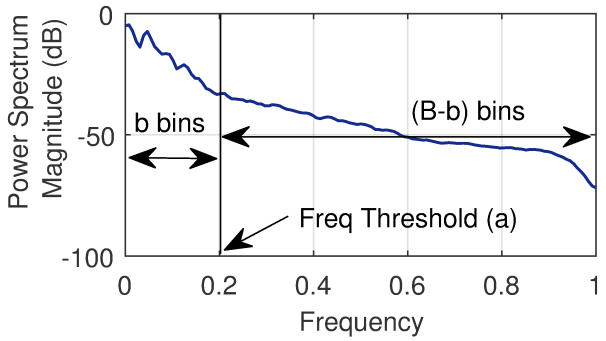


Fig. 12: Illustration of the basic idea of non-uniform binning of spectral energy in NBIP.

- **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 12. The optimal values of the parameters B, a, and b are empirically determined, which we will describe shortly.
- **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.

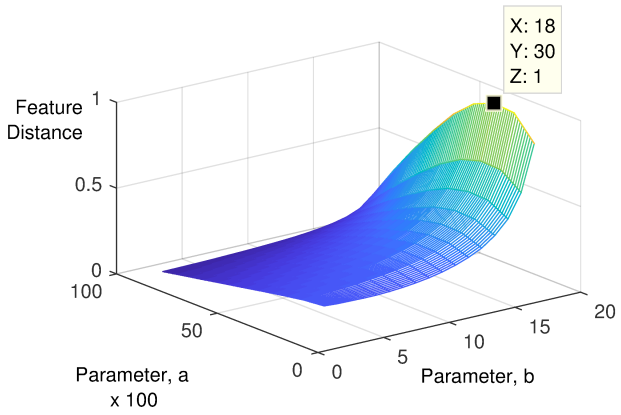
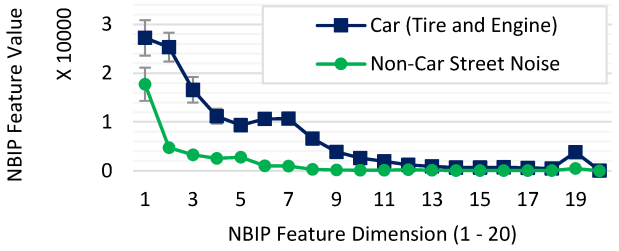


Fig. 13: NBIP search space for parameter optimization.

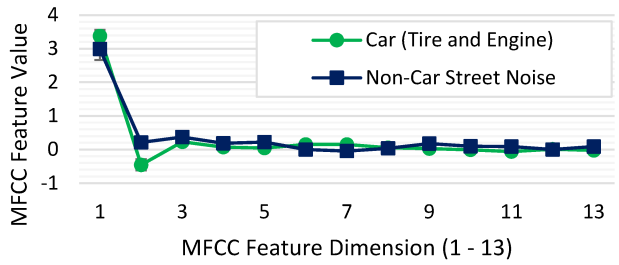
In order to find the optimum values of the parameters a, and b, we vary the parameters $0 \le a \le 1$ and $1 \le b \le B$ in small increments and compute the vector difference between features of car noises and all other non-car sounds. Figure 13 shows the search space for a and b for a fixed value 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 14 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 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 features described above are used to detect approaching cars/engine and tire noises only. As honk is not a noise like sound, we can not use our proposed feature NBIP in this scenario. So, we use MFCC in this case. For both types of classification (honks vs. engine/tire noises), we train separate Random Forest classifiers [7] which perform significantly better than other classifiers (e.g., Support Vector Machine [11]) that we applied on our data set.

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



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



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

Fig. 14: (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 NBIP features.

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 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 (which are computed 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. Platform Evaluation

A. Real-Time Performance

In this section we discuss the real-time performance of the system, the timing constraints involved, and how we designed our system to meet them. Response time is crucial for our system, as milliseconds can make a difference in saving the life of our user. 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 Figure 15. 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.

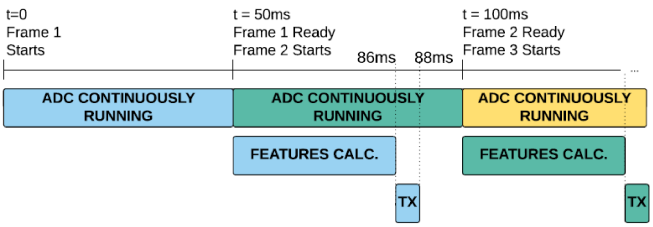


Fig. 15: Pipeline of the MCU processing. The "Features Calc." block represents all the operations involved in the features extraction, and "TX" represents the UART communication between the MCU and BLE module.

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 16. A button was 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 MCU and the BLE module firmware remain equivalent to the setup for standard operation. The average delay is on the order of 55ms as shown in Figure 17. 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.

The front-end hardware and the smartphone will have small physical distance as both of them will be on the user's body. This small distance will ensure very little effect on the connection due to the presence of multiple Bluetooth devices in the environment.

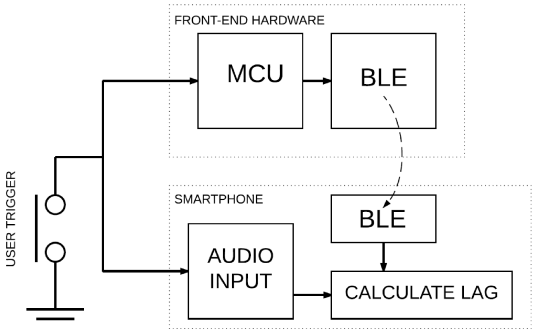


Fig. 16: Block diagram of the test setup for the latency between the features from the front-end hardware and the smartphone.

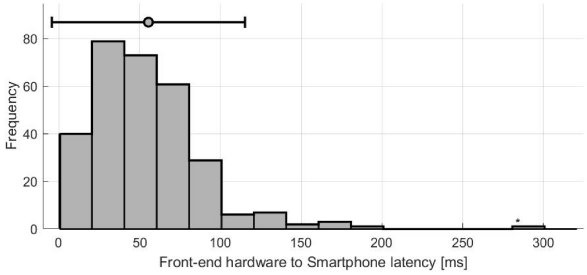


Fig. 17: Histogram of the front-end hardware to smartphone latency acquired with the Figure 16 test setup.

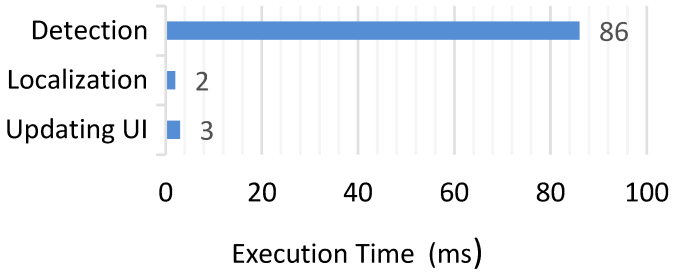


Fig. 18: Execution times of various components of the PAWS smartphone app.

Figure 18 shows the execution times of various components inside the smartphone application. The application runs four threads in parallel. Thread 1 is responsible for getting audio data using the single channel microphone. 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. The distance and direction estimators, which also runs in Thread 3, takes merely 2ms since these classifiers use precomputed features. The UI thread (Thread 4) takes 3ms to update the UI and to notify the user. The worst case execution time for the PAWS app is 91ms. Because we use a 50% overlap between successive windows, the PAWS app 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 PAWS 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 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 and 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 standard AAA Alkaline batteries. As shown in Table I,

TABLE I: Power Consumption and Price Breakdown

	Idle [mA]	Active [mA]	Unit Price [U$]
MCU (STM32f4)	4.37	50	3.20
BLE Transciever (nRF52)	0.46	7	6.40
MEMS Mics $\times 4$	0.48 $\times 4$	0.48 $\times 4$	0.40 $\times 4$
Amplifiers $\times 4$	2.34 $\times 4$	2.34 $\times 4$	1.60 $\times 4$
3.3V regulator	0.1	0.1	0.50
Total	16.21	68.4	18.10

TABLE II: CPU and Memory Footprint.

	CPU	Memory
STM32f4 (Active)	75.8%	84.916 KB
STM32f4 (Idle)	0%	6.908 KB
PAWS App (Active)	17.88%	32.58 MB

the total cost for the main electrical components is around U$18.00 per board.

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.

C. CPU and Memory Footprint

We measure the CPU and memory footprints of both the front-end data acquisition system as well as the smartphone application. The portion of the embedded front-end that consumes the most amount of resources (memory and CPU cycles) is the feature extraction process on the STM32f4.

Table II shows the average CPU and memory usages of the STM32f4 chip in PAWS. The CPU usage is almost 76% when the system is actively sampling and extracting features from audio sampled at 32kHz and in 50ms time slots. However when the system is idle, the CPU usage reaches 0%. This is because when the system is idle, the CPU of the STM32 and all of its main clocks are shut down; the system is only able to wake up again when it receives an external event from the BMD-300 BLE module, which is generated by the smartphone application on demand. Because of this, PAWS is able to save energy and CPU/memory resources when not actively in use.

V. Real-World Evaluation

To evaluate the end-to-end performance of the complete PAWS system in realistic settings, experiments were conducted in three environments with different characteristics: 1) a street inside a university campus; 2) by the side of a highway; and 3) in a metropolitan area.

A. Experimental Setup

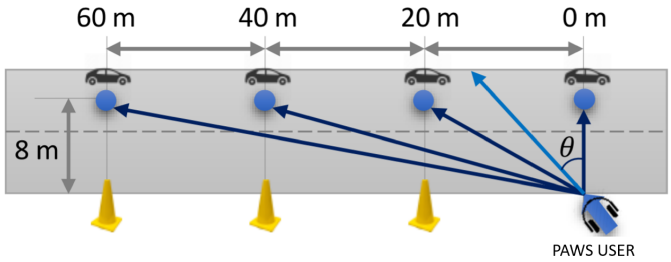


Fig. 19: Experiment scenario in campus street.

1. Campus street. The first experiment was done in a campus street with a speed limit of 25mph. We used three fixed markers (yellow cones) on the sidewalk and the PAWS app to evaluate the detection, direction, and distance accuracy of PAWS. Every time a vehicle passed a cone, a volunteer raised a flag and the event was logged in the original PAWS app. The setup is shown in Figure 19. 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.

TABLE III: Summary of Deployment Events.

Deployment	User (Facing Angle)	Honks	Car Events
Metro Area	0^o, $\pm45^o$, 180^o	48	165
Campus	0^o, $\pm45^o$, 90^o, $\pm135^o$, 180^o	0	97
Highway	0^o, $\pm45^o$, 90^o, $\pm135^o$, 180^o	0	65

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 highway is 45 mph as it is close to residential areas. As this experiment was done beside a highway, we were exposed to less pedestrian-borne noise but a heavy noise due to wind. 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 streets of Manhattan, New York City, where the number of cars, adjacent streets, and buildings in the surrounding area is very dense. Just as with the second experiment conducted near a highway, a time-stamped video was recorded and analyzed offline to obtain the ground truth. To evaluate the detection, direction, and distance classifiers, the classifier outputs were logged by PAWS and compared against the video 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 III provides some statistics of the deployment in all environments. For each one, the table shows how the PAWS user faced the road, and the number of logged honks and car events.

B. Results

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. Figure 20 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

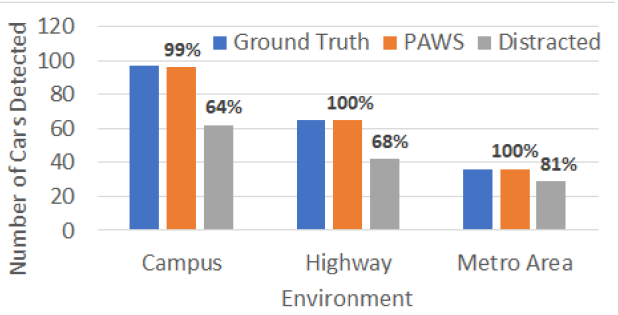


Fig. 20: Car detection performance.

is presented in Figure 21 for the metro area. The difference in the counts shown in Figure 21 and Figure 20 is that in Figure 20, 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 21 on the otherhand 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. However, we see from this figure that the number of false examples (no car cases) is around twenty times larger than the number of positive examples. This is because the amount of time we encounter a car when we are outside is much less than the amount of time when there is no car nearby. As such, the accuracy measure is largely dominated by the negative examples. Since the true negative rate is very high (95.2%), the accuracy of PAWS in the metropolitan area will also be very high regardless of the outcome of the true positive cases or situations where there is a car nearby. To get a better sense of accuracy, we use F1 score [1] for such a rare event detection system. The F1 score is calculated from true positive (TP), false positive (FP) and false negative (FN) using the formula $\frac{2\times TP}{(2\times TP)+FN+FP}$, whereas the regular definition of accuracy is measured with $\frac{TP+TN}{TP+TN+FN+FP}$.

We also compute the accuracy of distance and direction classification of PAWS and show the results in Figure 22 for all environments. We assume that the distances and directions reported by the ground truth collector is accurate. Each reported distance and direction is at first mapped to the corresponding distance level and direction class and then compared with the classification results of PAWS to compute the accuracy numbers. 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. There are also some cases where a user logged the location to an area close to the boundary between two classes while using the ground truth collector, which reduces computed accuracy.

Figure 23 shows the combined confusion matrix for all

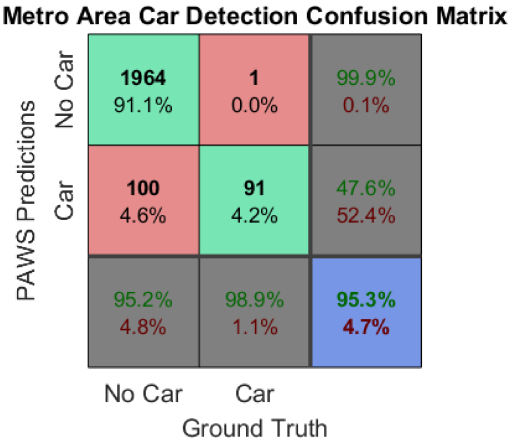


Fig. 21: Metro area car detection confusion matrix.

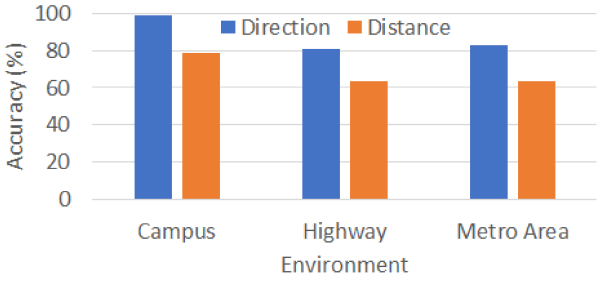


Fig. 22: Car localization accuracy.

distance predictions in all tested environments. 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 for distance estimation is 75.6%. Figure 24 breaks down the direction estimation result for different cones in 360^0, which has an average accuracy of 86.7%. The reason for these errors is that the participants naturally yaw their head about $\pm 22.5^0$ as they stand by the street. This effect could have been neutralized had we used an IMU to determine the staring angle of a user with respect to his body. We leave this enhancement of PAWS as a future work.

PAWS is able to alert a distracted user about an approaching car with around 98% accuracy. While the distance classification accuracy is around $63\% - 78\%$, as long as the majority of vehicles are correctly detected once they are within 60m, the user will have time to react. From Figure 24, we find that direction classifications for eight quadrants have varying degrees of accuracy. However, for most users, correctly determining left or right is sufficient, of which our system achieves between $94\% - 99\%$ accuracy depending on the environment.

C. Limitations and Future Work

The authors acknowledge that some scenarios can reduce the accuracy of the current system. Their effects on the predictions are detailed below.

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 are vastly different from the ones we have trained our system for. PAWS should be trained

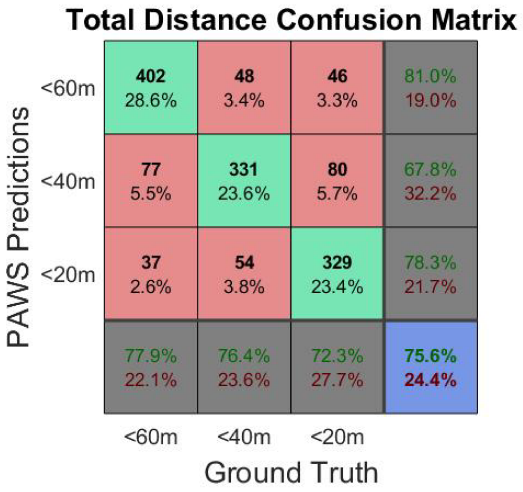


Fig. 23: confusion matrices for distance estimation.

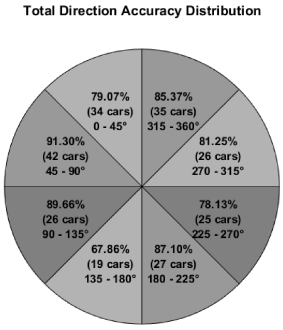


Fig. 24: Combined accuracy for PAWS directions prediction.

in as many scenarios as possible to be able to handle these new types of sounds.

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 at the same time can impair the localization of vehicles. PAWS predicts the location of the loudest source. However, the loudest source may not always be the most relevant vehicle to the user. Sound source separation and multiple sound source localization techniques are being investigated to improve the system.

Ultra-Low-Power Font-end Hardware An application specific integrated circuit (ASIC) is currently being developed to further reduce the power consumption of the front-end hardware. The ASIC will assist the system in the extraction of the acoustic features and will consume orders of magnitude less energy than the current approach. The current system will serve as a platform for evaluation and comparison of the ASIC based feature-extraction techniques combined with the machine-learning classification algorithms. The ultra-low-power ASIC will significantly relieve the computational burden of the system and allow enhancements to the algorithm.

VI. 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 [4], ultrasonic radars [3], and acoustic source localization [12]. In vehicular tracking, video based approaches have been widely used [37] [4] [22]. The amount of information that can be extracted from images is undoubtedly greater than any other types of sensors; it isn't by chance that humans learned to rely so much on their visual system. Commonality in vehicles' shapes and standardized road signs have enabled the use of sophisticated machine learning algorithms to identify and predict the movement of cars [39]. Although such systems offer outstanding solutions for devices that can be hosted in large platforms, e.g. in an autonomous car for collision prevention [32], these are not suitable for use in wearable systems. A major limitation is the computational requirements of real-time imaging processing and how feasible it is to develop a low-cost, power-efficient, fast-response product. Another major issue is the privacy of the user. As it was previously pointed out, having images of someone's activities being constantly taken reveals 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 [17] [9], 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 [6] [12] [34], PAWS uses machine learning algorithms to improve its predictions. By doing so, the system sacrifices resolution and 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 [15], gun shot detection [10], human activity (e.g. talking, crying, running etc) detection [2]. These works are mostly focused on identifying prominent sounds like gun shot or shouting rather than noise-like car sounds. [33] 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. [23] considered bus and trucks as events but had a very low accuracy of 24%. Other signals like video [29] [38] or seismic [13] signals have been used for vehicle detection. But they are not suitable for a wearable system like PAWS.

Other works for pedestrian safety have been done using other sensors like shoe sensors [16] or mobile app [36]. None of

these handles the possibility of the pedestrian listening to music or talking using headphones.

VII. Conclusion

This paper presents PAWS, 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, 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 distractions and dangers surround modern cities, innovative safety systems must and will arise as solutions to balance the common-citizen welfare.

References

[1] F1 score. https://en.wikipedia.org/wiki/Precision_and_recall.

[2] 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.

[3] B. Barshan and R. Kuc. A bat-like sonar system for obstacle localization. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(4), 1992.

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

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

[6] 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*, 99(6), 1996.

[7] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

[8] A. Champy. Google pixel budswireless headphones that help you do more, October 2017. [Online].

[9] 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.

[10] 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.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3), 1995.

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

[13] N. Evans. *Automated Vehicle Detection and Classification Using Acoustic and Seismic Signals*. PhD thesis, University of York, 2010.

[14] M. Galleso. Airpods: An easy guide to the best features. 2016.

[15] 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.

[16] 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*, MobiSys '15. ACM, 2015.

[17] 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*, 15(2), 2014.

[18] 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.

[19] S. G. Koolagudi and K. S. Rao. Emotion recognition from speech: a review. *International journal of speech technology*, 15(2), 2012.

[20] 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*, 1(3):70, 2017.

[21] 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*, volume 1. IEEE, 2001.

[22] 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*, 13(4), 2012.

[23] 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.

[24] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on*, 26(1), 1981.

[25] M. Park. Injuries while walking with headphones tripled, study finds, January 2012. [Online].

[26] C. J. Plack. The sense of hearing. 2005.

[27] 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.

[28] 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.

[29] 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.

[30] K. Shaver. Safety experts to pedestrians: Put the smartphones down and pay attention, September 2014. [Online].

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

[32] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2006.

[33] 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.

[34] 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*, 55(3), 2007.

[35] 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.

[36] 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*, HotMobile '12. ACM, 2012.

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

[38] 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.

[39] 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.