WalkSafe: A Pedestrian Safety App for Mobile Phone Users Who Walk and Talk While Crossing Roads

Tianyu Wang[1], Giuseppe Cardone[2], Antonio Corradi[2],
Lorenzo Torresani[1], and Andrew T. Campbell[1]
Computer Science Dartmouth College 1
6211 Sudikoff Lab, 03755 Hanover, NH, U.S.A.
{tianyuw, lorenzo, campbell}@cs.dartmouth.edu
University of Bologna 2
V.le Risorgimento, 2, 40136 Bologna, Italy
{giuseppe.cardone, antonio.corradi }@unibo.it

ABSTRACT

Research in social science has shown that mobile phone conversations distract users, presenting a significant impact to pedestrian safety; for example, a mobile phone user deep in conversation while crossing a street is generally more at risk than other pedestrians not engaged in such behavior. We propose *WalkSafe*, an Android smartphone application that aids people that walk and talk, improving the safety of pedestrian mobile phone users. WalkSafe uses the back camera of the mobile phone to detect vehicles approaching the user, alerting the user of a potentially unsafe situation; more specifically WalkSafe i) uses machine learning algorithms implemented on the phone to detect the front views and back views of moving vehicles and ii) exploits phone APIs to save energy by running the vehicle detection algorithm only during active calls. We present our initial design, implementation and evaluation of the WalkSafe App that is capable of real-time detection of the front and back views of cars, indicating cars are approaching or moving away from the user, respectively. WalkSafe is implemented on Android phones and alerts the user of unsafe conditions using sound and vibration from the phone. WalkSafe is available on Android Market.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time embedded systems

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
HotMobile'12 February 28–29, 2012, San Diego, CA, USA.
Copyright 2012 ACM 978-1-4503-1207-3 ...$10.00.

Keywords

Mobile Sensing System, Machine Learning, Mobile Phones, Pedestrian, Distraction

1. INTRODUCTION

Traffic accidents are a major public issue worldwide. In United States alone pedestrian fatalities account for 12% of all roadway deaths - 4092 in 2009 [7]. According to traffic safety statistics, car accidents that include pedestrians occur most frequently in urban areas where pedestrian activity and traffic volumes are much greater in comparison to rural areas. In terms of the location of accidents on roads, 65% of all crashes that involve pedestrians occur at non-intersections. Speeding, alcohol impairment and distraction are the main contributing factors in car crashes that involve pedestrian [11]. In most cases, drivers and pedestrians are equally responsible for accidents that occur.

The intuition that talking while walking across a road is dangerous is backed up by a large set of recent studies. A research trend is focusing on the influence of mobile phones on pedestrian safety because of the wide usage of phones as people move around during daily life. One early study on pedestrians crossing streets shows that mobile phone users exhibited more unsafe behavior than other pedestrians: that is, distracted walkers adopt less cautionary behavior (e.g., looking left and right, waiting for a walk signal) before crossing streets in comparison to non-distracted pedestrians [8, 10]. Experiments in a controlled virtual reality environment also reveal that when conversing on the phone pedestrians are less likely to recognize safe opportunities to cross the road [12].

A number of recent research projects demonstrate that wireless sensor network can enhance pedestrian safety, mostly focusing on intelligent transportation systems that help drivers to be aware of potentially unsafe conditions [2, 5, 6]; for example, communicating the location between pedestrians and vehicles using Dedicated Short Range Communications signals, helping vehicle drivers spot nearby pedestrians. In addition, lane, vehicle, and pedestrian detection technologies are used in auto-driven vehicles (e.g., Google cars) to enhance the safety of vehicles. However, to the best of our knowledge, our work is the first car detection classification and warning system implemented directly on off-the-shelf resource-limited mobile phone.

Existing works on intelligent transportation systems typically use sensors fixed on the vehicles to sense the surrounding environment and communicate information between pedestrians and vehicles; unfortunately such devices are still not ubiquitously available in the real world – and it is likely to be sometime before they are available because of cost and demands of scale. We take a different approach from these fixed infrastructure approaches and propose *WalkSafe*, a system that make pedestrians more aware of their surroundings while walking and talking by exploiting mobile sensors that are widely available, namely, smartphone cameras and accelerometers. Smartphones equipped with sensors [9] (e.g., accelerometer, gyroscope, GPS and camera) are becoming cheaper and more widespread, thus, we argue that smartphones are the perfect "sensor" carried by people that can be exploited to avoid road accidents by notifying users of any incoming cars during phone calls. However, sensing using smartphones presents several challenges; for example, a person lifting the phone produces noise in the accelerometer readings that must be filtered out before being able to use the accelerometer for pedestrians activity recognition. Similarly, if phone's camera is used for pedestrian safety, variance in light conditions, orientation of the phone with respect to cars and blurred video frames due to mobility make the use of the camera as a sensor to detect cars very challenging.

These challenges are barriers to implementing a robust car detection algorithms on the phone that can work under real-world conditions. A number of design decisions underpin the design of WalkSafe to seamlessly enhances pedestrians safety: i) WalkSafe should be immediately usable with minimum user dependency as possible; ii) WalkSafe should rely on the mobile phone as much as possible without the need for external resources; iii) WalkSafe should capable of fast real-time detection of cars at distance (e.g., 50 meters) in order to inform the user in a timely manner (e.g., a user has about 4 seconds to react to a car at 50 meters traveling at 30 mph) – the faster the detection system works the better for the user; and finally, iv) WalkSafe should preserve the phone user experience (e.g., it should not impact the performance of the phone or other applications).

In this paper, we present the design, implementation and evaluation of WalkSafe, the first system vehicle detection and pedestrian alert system for mobile phones. Typically, when a mobile user is speaking on the phone, the phone blocks the user's side view (either on the right or left side depending on which ear the user is using) while crossing the road. WalkSafe protects pedestrians distracted by phone conversations while crossing streets by using the phone's back camera to detect incoming vehicles, alerting the user via sound notifications and vibrations. The WalkSafe system uses the AdaBoost machine learning technique to train (offline) a car front view and rear view detection model, which is uploaded to the phone when the application is installed for real-time online classification (i.e., the detection of approaching cars). To improve the detection rate performance, WalkSafe preprocesses captured images to remove artifacts due to light conditions and phone orientation – WalkSafe solves these challenging environmental, user mobility and phone orientation issues in a robust manner. Finally, WalkSafe uses the Android APIs to trigger vehicle detection only when there are ongoing phone calls, thus saving battery lifetime, memory and computational resources.

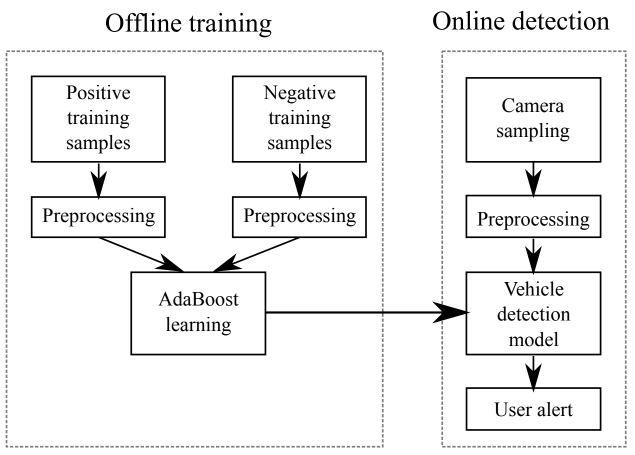

Figure 1: WalkSafe architecture. The vehicle recognition model is trained offline and then deployed on smartphones.

The paper is organized as follows. Section 2 details design of WalkSafe. Section 3 discusses the evaluation of the WalkSafe prototype application. We discuss related work in Section 4 and present some concluding remarks on future work in Section 5.

2. WALKSAFE DESIGN

The core WalkSafe car detection technology is based on image recognition algorithms. Image recognition is a computational intensive process that, if not carefully designed, can easily drain the computational resources and batteries of smartphones. To address this, WalkSafe bases its vehicle recognition process on a model that is first trained offline and then uploaded and used for the online vehicle recognition, as shown in Fig. 1. To build a effective vehicle recognition model, we prepare a dataset containing positive images (i.e., images that show the rear or frontal view of cars) and negative images (i.e., images that show side views of cars or random urban environments). Both sets of images are (i) preprocessed to normalize their features, for example, to normalize the brightness level and image size; then (ii) input to an algorithm that extracts characterizing features called "Haar-like features" [19] and then (iii) used to build a decision tree able to determine if a picture contains a front or rear view of a car. The resulting decision tree is then used by the WalkSafe application running on smartphones, running the online vehicle recognition in real-time.

The online vehicle recognition runs automatically whenever there is an ongoing phone call. WalkSafe activates the smartphone's camera and captures a picture of the surroundings. The picture is preprocessed to compensate for the phone tilt and illumination variations, and is then analyzed by the decision tree model built during the offline training phase, as discussed above. If the decision tree detects a car in the picture, it triggers an alert to warn the user of possible danger. In the following, we present a detailed discussion of the various steps of both the offline training and the online vehicle recognition on the phone.

2.1 Offline training

The offline training process is given a dataset of positive and negative image matches and builds a mathematical model that can be used to recognize positive matches online on the phone. Our offline training comprises four steps: (i) dataset building; (ii) training image preprocess – preprocessing the dataset to remove macroscopic heterogeneity between samples (such as, image size) and to artificially generate more samples; (iii) feature extraction from the images; and finally (iv) build the classification tree based on the features extracted. In what follows, we discuss each step in more detail.

The **dataset building** step is a fundamental step to building a good classifier; the number and quality of the training samples in the dataset has a great impact on the accuracy of the resulting classifier. In order to collect as many good quality positive training samples, we use the MIT CBCL car dataset, which contains 516 car front and rear view of cars, and the Caltech Cars dataset, which includes 526 images of rear view cars, for a total of 1032 positive samples [15, 3]. The images in the two datasets are shot by different cameras and their quality is different from the image quality shot when using the Google Nexus One camera. To address this issue, we preprocess the training images collected from the two dataset to resize them to the same resolution as the Google Nexus One phone camera. The negative training samples are fetched from Google Street View from around Hanover area. We use 3023 background images, which are selected to form the negative training set, containing no cars or only side view of cars.

The **training image preprocess** normalizes macroscopic characteristics of images (such as, image size and car position) and artificially generates additional image samples to improve the final classifier. We crop car regions from car image samples to leave as little background as possible in the positive samples and resize all images to 80x80 pixels. The cropped images are then transformed into gray scale images to reduce the impact of the colors of different cars. We normalized the gray scale intensity of each training images by the average intensity of the whole image dataset in order to equalize the light invariance of each training samples. The accuracy of the classifier is also affected by the alignment of the training dataset, thus, to minimize its impact, we aligned the cars in the training images to the bottom of cars. The width of the tires in each image are normalized to 65 pixels. Fig. 2 shows examples of positive training samples after preprocessing. To increase the robustness of the classifier, we expand the positive samples dataset by applying several image transformations. To make the classifier more robust in terms of light conditions and slightly different angles of the view of the cars, we apply 50 different gray scale intensity modifications (to reflect different light conditions experienced) and 0.1 degree distortion in x and y directions on each car image, thus, increasing the number of positive training images to 7000 samples with different light condition and slightly different view angle. The negative images are cropped from large size background images and transformed to gray scale. Some negative training images are shown in Fig. 3.

Finally, we use the **Haar feature-based cascade classifier driven by the AdaBoost algorithm** to build the car detection model. We use the OpenCV computer vision library to implement WalkSafe. The classifier building algorithm uses the positive and negative samples. We build a classifier that consists of several simpler classifiers (stages) based on Haar-like features. Haar-like features are a well-known technique that make feature extraction from images computationally easy. Haar-like features consider adjacent rectangular regions in a detection window, sum up the pixel intensities in these regions and calculates the difference between them – the resulting value is an Haar-like feature. The classifiers at every stage of the cascade pipeline are complex themselves being built out of basic classifiers, which "vote" on the classification of the same sample. The votes are weighted according to the Gentle AdaBoost algorithm, which is known to work well for this kind of applications [1, 19]. We train the cascade classifier on 20 stages. In each stage, a weak classifier is trained, in which a 0.999 hit rate and 0.5 false alarm rate are preserved. More specifically, we use a single layer of decision trees as the weak classifier in each stage. We use two kinds of decision trees as the weak classifier: one is the stump decision tree, which is a tree that has only one split node, and the other is Classification And Regression Tree (CART) with two split nodes. The accuracy of the two cascade classifiers are evaluated in the evaluation section, discussed later in the paper.

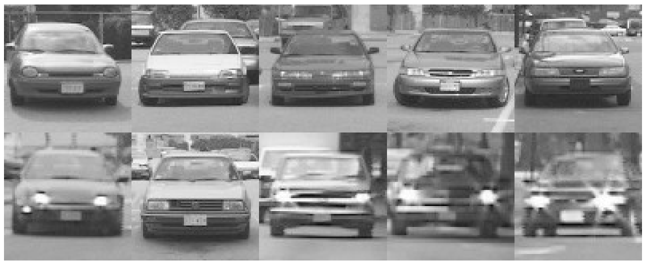
Figure 2: Positive training samples after preprocessing.

Figure 3: Negative training samples after preprocessing.

2.2 Online detection

The online car detection, which runs on the Andoid smartphone, comprises four steps: (i) image capture, (ii) image preprocess, (iii) car detection and (iv) alert dispatching. During the image capture step, WalkSafe captures a single image using the back facing camera on the smartphone.

Image capture is a CPU and I/O intensive activity, potentially impacting the phone's performance and battery lifetime. A possible solution is to delegate image processing to the cloud. However, this approach introduces high latency mainly because of limited cellular bandwidth. Such an approach to offload classification to the cloud will become more feasible as higher throughput becomes available. In our current version, WalkSafe triggers image capture on the phone only during active phone calls; that is, when it is actually needed, running the image processing algorithm locally on the phone. The car detection model built during the offline training phase can only recognize if a image represents a car and has little tolerance to background. To resolve this limitation, the captured image is subdivided using a scrolling window that moves by 3 pixels each time. Moreover, WalkSafe uses scrolling windows of different sizes, which allow the application to detect cars regardless of their apparent size. Each region of interest defined by scrolling windows is then preprocessed and checked by the learned classifier.

The **image preprocessing** step improves the classifier performance. WalkSafe uses the *accelerometer sensor data* to estimate the orientation of the mobile phone, and aligns the test image according to the direction of gravity. Hence, the bottom of the cars are parallel to the bottom of test images. This allows WalkSafe to correctly detect cars even if the mobile phone is arbitrarily rotated, which is the typical case when users speak on the phone – people hold the phone at different orientations and the WalkSafe system is to be adaptive to this real-world behavior. In addition, WalkSafe also performs gray scale intensity normalization of each test image to eliminate the illumination difference between the test image and the images of the training dataset. The advantage of using a classifier based on Haar-like features, is that it can be easily scaled to detect objects of interest in subregions of the test images without resizing the image itself – which, is more efficient than resizing the image itself. Thus, WalkSafe repeatedly scans the captured image analyzing windows of different sizes in different places.

After preprocessing, the test images are input to the **car detection** step, which uses the classification model built during the offline training. The classifier is designed to run in real time, as it can refuse the negative images very quick if the test image can not pass a stage. Only if a region of interest passes all stages does WalkSafe define that region as a car and then proceeds to dispatch a notification to the user. In the current implementation the **user alert** is a vibration, which notifies the user about the incoming car.

3. WALKSAFE EVALUATION

In this section, we discuss initial results from the evaluation of the car detection model and the WalkSafe prototype. A near full version of the WalkSafe App can be downloaded from the Market. We implemented WalkSafe application for Android 2.3.4 operating system and tested it on Google Nexus One, which is equipped with 1 GHz Qualcomm Snapdragon QSD8250 CPU, has 512 MiB RAM memory, an accelerometer (which WalkSafe uses to rotate images) and a back facing camera capable of video capture at 20 frames per second with 720x480 pixel resolution. The WalkSafe prototype is implemented in Java and leverages the OpenCV (Open Source Computer Vision) library for computer vision algorithms [1]. In what follows, we describe the performance of two Haar cascade car detection models with different training parameters. We also report the accuracy of the WalkSafe prototype used in a real world evaluation, as well as system performance bench marks for computation time, CPU and memory usage and battery consumption.

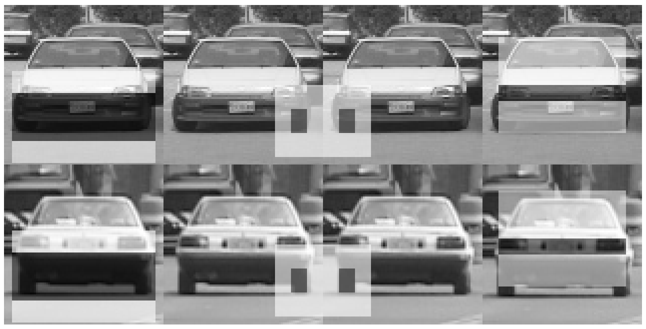
Figure 4: Examples of Haar-like features extracted from the car dataset.

To evaluate the performance of our car detection models, we select 48 cropped car front view and rear view images and 300 different Google Street View images from the training dataset, and generated 216 testing images with car location annotated in the images. Two Haar cascade car detection models are trained by Gentle AdaBoost algorithm as discussed earlier. One car detection model is trained using a simple tree stump in each weak classifier, while the other model is trained using the CART decision tree with two splits in each weak classifier. We train the Haar cascade car detection model using 20x20 positive pattern size. 0.999 hit rate and 0.5 false positive rate are set in each weak classifier training phase. We apply the two car detection models to the same test dataset. The ROC curves for each of two models are shown in Fig. 5. The ROC curves shows the Haar cascade car detection model which consists of tree stumps as weak classifier performs better than the one that consists of two-split CART decision trees. Therefore, we choose the car detection model with simple tree stumps as each weak classifier in the WalkSafe prototype. The first four features selected by the Gentle AdaBoost algorithms are shown in Fig. 4. As we can see from the figure the car front view and rear view training images, the bumper part of the car is selected as the most discriminative feature. The right tire and the left tire are selected as the second and third discriminative features, follow by the headlights and backlights.

To evaluate the practical viability of WalkSafe, we carried out a number of real world experiments: 30-minute car detection experiments were performed from 13:00 to 17:00 hours at five different locations in Hanover, NH. The mobile phone was held by a pedestrian standing at the side of the street. We recorded the video captured by a Nexus One phone, and counted the number of cars that appeared in the video as ground truth. We compared the ground truth to the number of cars detected by the WalkSafe prototype. The average results are shown in Table 1. From the experiments, we also measure the approximately maximum detection distance. WalkSafe was able to detect cars 50 meters away from the pedestrians. Fig. 6 and 7 show some true positive and false positive detections. A small portion of false positive results are the side view of the vehicles. However, the false positive rate is very low in the real world experiments (as it is shown in Table 1) so that the normal user experience can

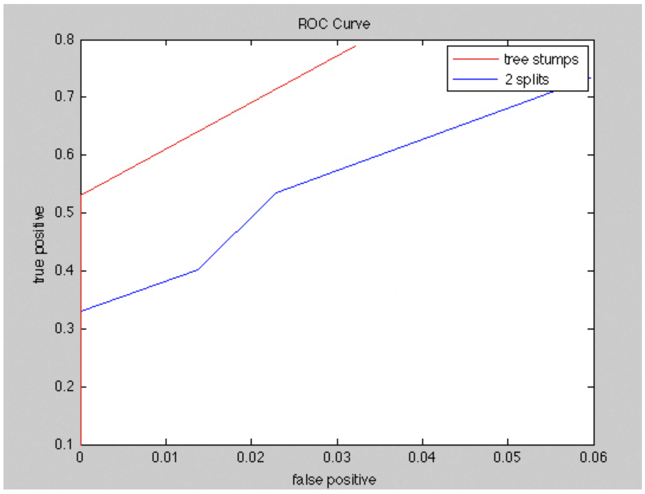

Figure 5: Car detection ROC Curve. The red line is the ROC curve of tree stumps and the blue line is the ROC curve of two-split CART decision trees.

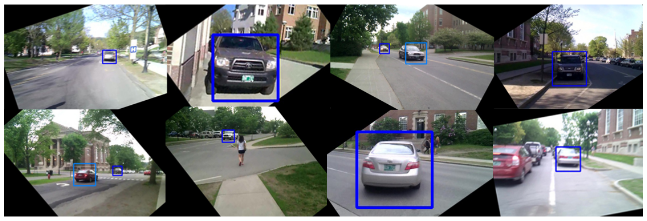

Figure 6: True positive detections in real world scenario using WalkSafe on the phone with different user phone orientation.

be preserved when pedestrian phone users are walking along the street on the sidewalk. The high rate of correctly detect cars makes us confident that WalkSafe is very promising solution, indicating that WalkSafe is a workable, practical and robust approach to detecting on coming cars under real-world conditions (e.g., with different cars, users, user phone orientation, user mobility and light conditions).

We also evaluated the system performance of WalkSafe. The Nexus One's rear camera is capable of capturing up to 20 frames per second (FPS). WalkSafe uses approximately 140 milliseconds to infer the car position in one frame, which means that WalkSafe system is capable of processing about 8 FPS. In our real world experiments, WalkSafe detects most of the cars coming towards the pedestrians. However, under certain scenarios WalkSafe only detected cars when they were very close to the pedestrians, limiting the time for the pedestrians to react safely. Improvements about this aspect

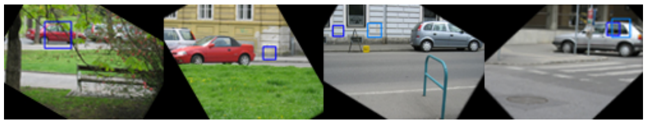

Figure 7: False positive detections in real world scenario using WalkSafe on the phone with different user phone orientation.

Table 1: 30-minute WalkSafe car detection experiments in real world

Coming car direction	Front view	Rear view
Number of cars detected	84	57
Number of cars missed	25	18
True positive rate	77%	76%
Number of false positive	3	
Maximum detection distance	50 meters	

can be expected by leveraging faster phone processors and additional quicker inference pipelines, that we are studying as part of our future work.

WalkSafe is quite lightweight in terms of RAM usage on Nexus One requiring only 3.95 MB. However, the WalkSafe prototype demands high computational resources taking up to 12% of CPU usage. After running WalkSafe for 5 hours on the Nexus One, the battery level dropped to 15% after a full charge. However, it should be noted that in a real world usage, WalkSafe is only activated during phone calls, thus, alleviating the energy drain issue.

4. RELATED WORK

There is a growing interest in using smartphones, short-range communication systems and computer vision techniques for pedestrian safety. In what follows, we discuss the related work. There are several commercial products, (e.g., “Text and Walk” and “Walk 'n Email”) that leverage the smartphone's back facing camera to let users write SMS and e-mails while walking safely by displaying the road ahead as application background. However, [14] shows that these applications can help, but people are overloaded and therefore somewhat limited in concurrent multitasks processing, thus, users may not be aware of dangers even if they are displayed as application background. In [4], the authors discuss the response time for smartphone applications that warn users of collision risks, analyzing approaches based on a centralized server and on ad-hoc communication connections between cars and pedestrians. The Oki Electric company has developed a Dedicated Short Range Communication (DSRC) interface for mobile phones that allows drivers to notify nearby vehicles and pedestrians of their location from the embedded GPS [13]. It should be noted that these approach depend on cooperation of vehicles, network availability and specialized hardware, thus, making a large scale deployment difficult, costly, and perhaps, unlikely. In [5], the authors present a survey of recent research car and pedestrian detection techniques used by a number of systems – however, none applied to the smartphone scenario addressed by WalkSafe. Car and pedestrian detection systems can rely on a wide range of sensors (e.g., visible light imaging, infrared imaging, time of flight sensors, RADARs, LASER scanners), video cues (e.g., shape, motion, depth) and classifiers (e.g., AdaBoost, neural networks, cascade of classifiers). In [18], the authors describe a car detection system based on Haar features that exploits the on-board cameras available in some cars. [16] describes a complete framework based on Haar features, enhanced by a tracking algorithm, to continuously track vehicles. This system performs very well, but it is implemented using on board car cameras and relies on external hardware support, which is ill-fitting to resource-constrained smartphones. [17] is a project that aims at improving the safety of bikers by detecting approaching cars using computer vision algorithms on a dedicated hardware platform.

5. CONCLUSION

In this paper, we have presented the design and initial prototyping and evaluation of the WalkSafe App for people that walk and talk and cross roads. WalkSafe operates at the intersection of vision, learning and mobile computing – solving a number of real-world problems presented by the mobility of the user, phone orientation, different lightening conditions and implementation on a resource limited phone. To the best of our knowledge WalkSafe is the first smartphone app that uses the camera and accelerometer sensors and classification pipeline to alert users of unsafe cars approaching them. WalkSafe is available on Android Market[1].

Our future work includes three main directions. First, we plan to extend the training dataset, improving the vehicle detection model and reduce energy consumption. We aim to increase the type of vehicles that WalkSafe can detect including buses, trucks, push bikes and motor bikes. We also aim to study car detection at night, which represents the most dangerous period of the day for pedestrians (i.e., 47% of the fatalities [11] occur between the hours of 18:00 and midnight in the U.S.A). Our efforts to enhance the vehicle detection model is twofold: speeding up the recognition algorithm and adding support for continuous tracking. To speed up the recognition algorithm, we will limit the vehicle search for a specific frame in areas surrounding vehicles detected in the previous frame. Reducing the time to process a frame will make it easier to implement continuous tracking, allowing us to distinguish between moving vehicles and parked ones. As the processing capability of phones increases the real-time performance of WalkSafe will also benefit. Finally, we plan integrate activity classification that exploits the accelerometer on the phone to automatically disable WalkSafe when the user is not walking.

6. REFERENCES

[1] G. Bradski and A. Kaehler. *Learning OpenCV.* O'Reilly Media, 2008.

[2] F. Bu and C.-Y. Chan. Pedestrian detection in transit bus application: sensing technologies and safety solutions. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 100 – 105, June 2005.

[3] Computational Vision Group at Caltech. Cars 2001 (rear), 2001. http://www.vision.caltech.edu/html-files/archive.html.

[4] K. David and A. Flach. Car-2-x and pedestrian safety. *Vehicular Technology Magazine, IEEE*, 5(1):70 –76, March 2010.

[5] T. Gandhi and M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):413 –430, September 2007.

[6] D. Gavrila. Sensor-based pedestrian protection. *IEEE Intelligent Systems*, 16(6):77 – 81, 2001.

[7] Governors Highway Safety Association. Pedestrian traffic fatalities by state: 2010 preliminary data, http://www.ghsa.org, 2010.

[8] J. Hatfield and S. Murphy. The effects of mobile phone use on pedestrian crossing behaviour at signalised and unsignalised intersections. *Accident Analysis & Prevention*, 39(1):197 – 205, 2007.

[9] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[10] J. Nasar, P. Hecht, and R. Wener. Mobile telephones, distracted attention, and pedestrian safety. *Accident Analysis & Prevention*, 40(1):69 – 75, 2008.

[11] National Highway Traffic Safety Administration. Traffic safety facts 2009. Washington, DC, 2010.

[12] M. B. Neider, J. S. McCarley, J. A. Crowell, H. Kaczmarski, and A. F. Kramer. Pedestrians, vehicles, and cell phones. *Accident Analysis & Prevention*, 42(2):589 – 594, 2010.

[13] Oki Electric Industry. Oki succeeds in trial production of world's first "safety mobile phone" to improve pedestrian safety, May 2007. http://www.oki.com/en/press/2007/z07023e.html.

[14] E. Ophir, C. Nass, and A. D. Wagner. Cognitive control in media multitaskers. *Proc. of the National Academy of Sciences*, 106(37):15584–15587, 2009.

[15] C. Papageorgiou and T. Poggio. A trainable object detection system: Car detection in static images. Technical Report 1673, October 1999. (CBCL Memo 180).

[16] S. Sivaraman and M. Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267 –276, June 2010.

[17] S. Smaldone, C. Tonde, V. K. Ananthanarayanan, A. Elgammal, and L. Iftode. The cyber-physical bike: A step towards safer green transportation. In *HotMobile '11: Proc. of the 12th Workshop on Mobile Computing Systems and Applications*, March 2011.

[18] Z. Sun, G. Bebis, and R. Miller. Monocular precrash vehicle detection: features and classifiers. *IEEE Transactions on Image Processing*, 15(7):2019 –2034, July 2006.

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:511, 2001.

[1]The current version of WalkSafe available on the Android Market does not run as background service because Android has currently little support for non GUI applications, such as WalkSafe that access the phone's camera; thus, the fully functional app can only correctly run on a customized Android version. We are addressing this issue and plan to publish WalkSafe as a background service in the near future. The version of WalkSafe on the Market is almost fully functional in terms demonstrating the core technology of real-time car detection using sensor data from the camera and accelerometer, as discussed in this paper. It can be downloaded from https://market.android.com/details?id=edu.dartmouth.cs.walksafe