EyePhone: Activating Mobile Phones With Your Eyes

Emiliano Miluzzo, Tianyu Wang, Andrew T. Campbell

Computer Science Department
Dartmouth College
Hanover, NH, USA

ABSTRACT

As smartphones evolve researchers are studying new techniques to ease the human-mobile interaction. We propose *EyePhone*, a novel "hand-free" interfacing system capable of driving mobile applications/functions using only the user's eyes movement and actions (e.g., wink). EyePhone tracks the user's eye movement across the phone's display using the camera mounted on the front of the phone; more specifically, machine learning algorithms are used to: i) track the eye and infer its position on the mobile phone display as a user views a particular application; and ii) detect eye blinks that emulate mouse clicks to activate the target application under view. We present a prototype implementation of EyePhone on a Nokia N810, which is capable of tracking the position of the eye on the display, mapping this positions to an application that is activated by a wink. At no time does the user have to physically touch the phone display.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation, Human Factors, Measurement, Performance

Keywords

Human-Phone Interaction, Mobile Sensing Systems, Machine Learning, Mobile Phones

1. INTRODUCTION

Human-Computer Interaction (HCI) researchers and phone vendors are continuously searching for new approaches to reduce the effort users exert when accessing applications on limited form factor devices such as mobile phones. The most significant innovation of the past few years is the adoption of touchscreen technology introduced with the Apple iPhone [1] and recently followed by all the other major vendors, such as Nokia [2] and HTC [3]. The touchscreen has changed the way people interact with their mobile phones because it provides an intuitive way to perform actions using the movement of one or more fingers on the display (e.g., pinching a photo to zoom in and out, or panning to move a map).

Several recent research projects demonstrate new people-to-mobile phone interactions technologies [4, 5, 6, 7, 8, 9, 10]. For example, to infer and detect gestures made by the user, phones use the on-board accelerometer [4, 7, 8], camera [11, 5], specialized headsets [6], dedicated sensors [9] or radio features [10]. We take a different approach than that found in the literature and propose the *EyePhone* system which exploits the eye movement of the user captured using the phone's front-facing camera to trigger actions on the phone.

HCI research has made remarkable advances over the last decade [12] facilitating the interaction of people with machines. We believe that *human-phone interaction (HPI)* extends the challenges not typically found in HCI research, more specially related to the phone and how we use it. We term HPI as developing techniques aimed at advancing and facilitating the interaction of people with mobile phones. HPI presents challenges that differ somewhat from traditional HCI challenges. Most HCI technology addresses the interaction between people and computers in "ideal" environments, i.e., where people sit in front of a desktop machine with specialized sensors and cameras centered on them. In contrast, mobile phones are mobile computers with which people interact on the move under varying conditions and context. Any phone's sensors, e.g., accelerometer, gyroscope, or camera, used in a HPI technology must take into account the constraints that mobility brings into play. For example, a person walking produces a certain signature in the accelerometer readings that must be filtered out before being able to use the accelerometer for gesture recognition (e.g., double tapping the phone to stop an incoming phone call). Similarly, if the phone's camera is adopted in a HPI application [11, 5, 9] the different light conditions and blurred video frames due to mobility make the use of the camera to infer events very challenging. For these reasons HCI technologies need to be extended to be applicable to HPI environments.

In order to address these goals HPI technology should be less intrusive; that is, i) it should not rely on any external devices other than the mobile phone itself; ii) it should be readily usable with minimum user dependency as possible; iii) it should be fast in the inference phase; iv) it should be

MobiHeld 2010, August 30, 2010, New Delhi, India.

lightweight in terms of computation; and v) it should preserve the phone user experience, e.g., it should not deplete the phone battery over normal operations.

We believe that HPI research advances will produce a leap forward in the way people use their mobile phones by improving people safety, e.g., HPI techniques should aim to reduce the distraction and consequently the risk of accidents if driving for example, or facilitating the use of mobile phones for impaired people (e.g., people with disabilities).

We propose EyePhone, the first system capable of tracking a user's eye and mapping its current position on the display to a function/application on the phone using the phone's front-facing camera. EyePhone allows the user to activate an application by simply "blinking at the app", emulating a mouse click. While other interfaces could be used in a hand-free manner, such as voice recognition, we focus on exploiting the eye as a driver of the HPI. We believe EyePhone technology is an important alternative to, for example, voice activation systems based on voice recognition, since the performance of a voice recognition system tends to degrade in noisy environments.

The front camera is the only requirement in EyePhone. Most of the smartphones today are equipped with a front camera and we expect that many more will be introduced in the future (e.g., Apple iPhone 4G [1]) in support of video conferencing on the phone. The EyePhone system uses machine learning techniques that after detecting the eye create a template of the open eye and use template matching for eye tracking. Correlation matching is exploited for eye wink detection [13]. We implement EyePhone on the Nokia N810 tablet and present experimental results in different settings. These initial results demonstrate that EyePhone is capable of driving the mobile phone. An EyePhone demo can be found at [15].

The paper is organized as follows. In Section 2, we discuss the challenges encountered in the development of HPI technology. Section 3 presents the design of the EyePhone system followed by its evaluation in Section 4. The future research direction are reported in Section 5. Section 6 discusses related work and Section 7 finishes with some concluding remarks.

2. HUMAN-PHONE INTERACTION

Human-Phone Interaction represents an extension of the field of HCI since HPI presents new challenges that need to be addressed specifically driven by issues of mobility, the form factor of the phone, and its resource limitations (e.g., energy and computation). More specifically, the distinguishing factors of the mobile phone environment are *mobility* and the *lack of sophisticated hardware support*, i.e., specialized headsets, overhead cameras, and dedicated sensors, that are often required to realize HCI applications. In what follows, we discuss these issues.

Mobility Challenges. One of the immediate products of mobility is that a mobile phone is moved around through unpredicted context, i.e., situations and scenarios that are hard to see or predict during the design phase of a HPI application. A mobile phone is subject to uncontrolled movement, i.e., people interact with their mobile phones while stationary, on the move, etc. It is almost impossible to predict how and where people are going to use their mobile phones. A HPI application should be able to operate reliably in any encountered condition. Consider the following examples: two HPI applications, one using the accelerometer, the other relying on the phone's camera. Imagine exploiting the accelerometer to infer some simple gestures a person can perform with the phone in their hands, e.g., shake the phone to initiate a phone call, or tap the phone to reject a phone call [7]. What is challenging is being able to distinguish between the gesture itself and any other action the person might be performing. For example, if a person is running or if a user tosses their phone down on a sofa, a sudden shake of the phone could produce signatures that could be easily confused with a gesture. There are many examples where a classifier could be easily confused. In response, erroneous actions could be triggered on the phone. Similarly, if the phone's camera is used to infer a user action [5][9], it becomes important to make the inference algorithm operating on the video captured by the camera robust against lighting conditions, which can vary from place to place. In addition, video frames blur due to the phone movement. Because HPI application developers cannot assume any optimal operating conditions (i.e., users operating in some idealized manner) before detecting gestures in this example, (e.g., requiring a user to stop walking or running before initiating a phone call by a shaking movement), then the effects of mobility must be taken into account in order for the HPI application to be reliable and scalable.

Hardware Challenges. As opposed to HCI applications, any HPI implementation should not rely on any external hardware. Asking people to carry or wear additional hardware in order to use their phone [6] might reduce the penetration of the technology. Moreover, state-of-the art HCI hardware, such as glass mounted cameras, or dedicated helmets are not yet small enough to be conformably worn for long periods of time by people. Any HPI application should rely as much as possible on just the phone's on-board sensors.

Although modern smartphones are becoming more computationally capable [16], they are still limited when running complex machine learning algorithms [14]. HPI solutions should adopt lightweight machine learning techniques to run properly and energy efficiently on mobile phones.

3. EYEPHONE DESIGN

One question we address in this paper is how useful is a cheap, ubiquitous sensor, such as the camera, in building HPI applications. We develop eye tracking and blink detection mechanisms based algorithms [13, 17] originally designed for desktop machines using USB cameras. We show the limitations of an off-the-shelf HCI technique [13] when used to realize a HPI application on a resource limited mobile device such as the Nokia N810. The EyePhone algorithmic design breaks down into the following pipeline phases: 1) an eye detection phase; 2) an open eye template creation phase; 3) an eye tracking phase; 4) a blink detection phase. In what follows, we discuss each of the phases in turn.

Eye Detection. By applying a motion analysis technique which operates on consecutive frames, this phase consists on finding the contour of the eyes. The eye pair is identified by the left and right eye contours. While the original algorithm [17] identifies the eye pair with almost no error when running on a desktop computer with a fixed camera (see the left image in Figure 1), we obtain errors when the algorithm is implemented on the phone due to the quality of the N810 camera compared to the one on the desktop and

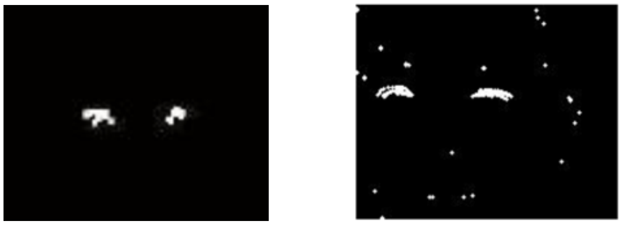

Figure 1: Left figure: example of eye contour pair returned by the original algorithm running on a desktop with a USB camera. The two white clusters identify the eye pair. Right figure: example of number of contours returned by EyePhone on the Nokia N810. The smaller dots are erroneously interpreted as eye contours.

the unavoidable movement of the phone while in a person's hand (refer to the right image in Figure 1). Based on these experimental observations, we modify the original algorithm by: i) reducing the image resolution, which according to the authors in [13] reduces the eye detection error rate, and ii) adding two more criteria to the original heuristics that filter out the false eye contours. In particular, we filter out all the contours for which their width and height in pixels are such that $width_{min} \leq$ width $\leq width_{max}$ and $height_{min} \leq$ height $\leq height_{max}$. The $width_{min}$, $width_{max}$, $height_{min}$, and $height_{max}$ thresholds, which identify the possible sizes for a true eye contour, are determined under various experimental conditions (e.g., bright, dark, moving, not moving) and with different people. This design approach boosts the eye tracking accuracy considerably, as discussed in Section 4.

Open Eye Template Creation. While the authors in [13] adopt an online open eye template creation by extracting the template every time the eye pair is lost (this could happen because of lighting condition changes or movement in the case of a mobile device), EyePhone does not rely on the same strategy. The reduced computation speed compared to a desktop machine and the restricted battery requirements imposed by the N810 dictate a different approach. EyePhone creates a template of a user's open eye once at the beginning when a person uses the system for the first time using the eye detection algorithm described above[1]. The template is saved in the persistent memory of the device and fetched when EyePhone is invoked. By taking this simple approach, we drastically reduce the runtime inference delay of EyePhone, the application memory footprint, and the battery drain. The downside of this off-line template creation approach is that a template created in certain lighting conditions might not be perfectly suitable for other environments. We intend to address this problem as part of our future work.

In the current implementation the system is trained individually, i.e., the eye template is created by each user when the application is used for the first time. In the future, we will investigate eye template training by relying on pre-collected data from multiple individuals. With this supervised learning approach users can readily use EyePhone without going through the initial eye template creation phase.

Eye Tracking. The eye tracking algorithm is based on template matching. The template matching function calculates a correlation score between the open eye template, created the first time the application is used, and a search window. In order to reduce the computation time of the template matching function and save resources, the search window is limited to a region which is twice the size of a box enclosing the eye. These regions are shown in Figure 2, where the outer box around the left eye encloses the region where the correlation score is calculated. The correlation coefficient we rely on, which is often used in template matching problems, is the normalized correlation coefficient defined in [18]. This coefficient ranges between -1 and 1. From our experiments this coefficient guarantees better performance than the one used in [13]. If the normalized correlation coefficient equals 0.4 we conclude that there is an eye in the search window. This threshold has been verified accurate by means of multiple experiments under different conditions (e.g., bright, dark, moving, not moving).

Blink Detection. To detect blinks we apply a thresholding technique for the normalized correlation coefficient returned by the template matching function as suggested in [13]. However, our algorithm differs from the one proposed in [13]. In [13] the authors introduce a single threshold T and the eye is deemed to be open if the correlation score is greater than T, and closed vice versa. In the EyePhone system, we have two situations to deal with: the quality of the camera is not the same as a good USB camera, and the phone's camera is generally closer to the person's face than is the case of using a desktop and USB camera. Because of this latter situation the camera can pick up iris movements, i.e., the interior of the eye, due to eyeball rotation. In particular, when the iris is turned towards the corner of the eye, upwards or downwards, a blink is inferred even if the eye remains open. This occurs because in this case the majority of the eye ball surface turns white which is confused with the color of the skin. We derive four thresholds: T_1^{min} = 0.64, T_1^{max}= 0.75, T_2^{min} = -0.53, and T_2^{max} = -0.45. These

Figure 2: Eye capture using the Nokia N810 front camera running the EyePhone system. The inner white box surrounding the right eye is used to discriminate the nine positions of the eye on the phone's display. The outer box encloses the template matching region.

[1]The eye template is created by putting the phone at a distance of about 20 cm from the eyes.

Table 1: EyePhone average eye tracking accuracy for different positions of the eye in different lighting and movement conditions and blink detection average accuracy. Legend: DS = eye tracking accuracy measured in daylight exposure and being steady; AS = eye tracking accuracy measured in artificial light exposure and being steady; DM = eye tracking accuracy measured in daylight exposure and walking; BD = blink detection accuracy in daylight exposure.

Eye position	DS	AS	DM	BD
Top left	76.73%	74.50%	82.81%	84.14%
Top center	79.74%	97.78%	79.16%	78.47%
Top right	80.35%	95.06%	60%	82.17%
Middle left	98.46%	97.19%	70.99%	74.72%
Middle center	99.31%	84.09%	76.52%	79.55%
Middle right	99.42%	75.79%	65.15%	80.1%
Bottom left	98.36%	93.22%	78.83%	74.53%
Bottom center	90.76%	71.46%	85.26%	67.41%
Bottom right	84.91%	93.56%	78.25%	72.89%

thresholds are determined experimentally and again under different experimental conditions as discussed previously. If we indicate with c_{min} and c_{max}, respectively, the minimum and maximum normalized correlation coefficient values returned by the template matching function, the eye is inferred to be closed if $T_1^{min} \leq c_{max} \leq T_1^{max}$ and $T_2^{min} \leq c_{min} \leq T_2^{max}$. It is inferred to be open otherwise. The results of the blink detection algorithm are discussed in Section 4.

4. EVALUATION

In this section, we discuss initial results from the evaluation of the EyePhone prototype. We implement EyePhone on the Nokia N810 [19]. The N810 is equipped with a 400 MHz processor and 128 MB of RAM[2]. The N810 operating system is Maemo 4.1, a Unix based platform on which we can install both the C OpenCV (Open Source Computer Vision) library [20] and our EyePhone algorithms which are cross compiled on the Maemo scratchbox. To intercept the video frames from the camera we rely on GStreamer [21], the main multimedia framework on Maemo platforms. In what follows, we first present results relating to average accuracy for eye tracking and blink detection for different lighting and user movement conditions to show the performance of EyePhone under different experimental conditions. We also report system measurements, such as CPU and memory usage, battery consumption and computation time when running EyePhone on the N810. All experiments are repeated five times and average results are shown.

Daylight Exposure Analysis for a Stationary Subject. The first experiment shows the performance of EyePhone when the person is exposed to bright daylight, i.e., in a bright environment, and the person is stationary. The eye tracking results are shown in Figure 2. The inner white box in each picture, which is a frame taken from the front camera when the person is looking at the N810 display while holding the device in their hand, represents the eye position on the phone display. It is evident that nine different positions for the eye are identified. These nine positions of the eye can be mapped to nine different functions and applications as shown in Figure 4. Once the eye locks onto a position (i.e., the person is looking at one of the nine buttons on the display), a blink, acting as a mouse click, launches the application corresponding to the button. The accuracy of the eye tracking and blink detection algorithms are reported in Table 1. The results show we obtain good tracking accuracy of the user's eye. However, the blink detection algorithms accuracy oscillates between ∼67 and 84%. We are studying further improvements in the blink detection as part of future work.

Artificial Light Exposure for a Stationary Subject. In this experiment, the person is again not moving but in an artificially lit environment (i.e., a room with very low daylight penetration from the windows). We want to verify if different lighting conditions impact the system's performance. The results, shown in Table 1, are comparable to the daylight scenario in a number of cases. However, the accuracy drops. Given the poorer lighting conditions, the eye tracking algorithm fails to locate the eyes with higher frequency.

Daylight Exposure for Person Walking. We carried out an experiment where a person walks outdoors in a bright environment to quantify the impact of the phone's natural movement; that is, shaking of the phone in the hand induced by the person's gait. We anticipate a drop in the accuracy of the eye tracking algorithm because of the phone movement. This is confirmed by the results shown in Table 1, column 4. Further research is required to make the eye tracking algorithm more robust when a person is using the system on the move.

Impact of Distance Between Eye and Tablet. Since in the current implementation the open eye template is created once at a fixed distance, we evaluate the eye tracking performance when the distance between the eye and the tablet is varied while using EyePhone. We carry out the measurements for the middle-center position in the display (similar results are obtained for the remaining eight positions) when the person is steady and walking. The results

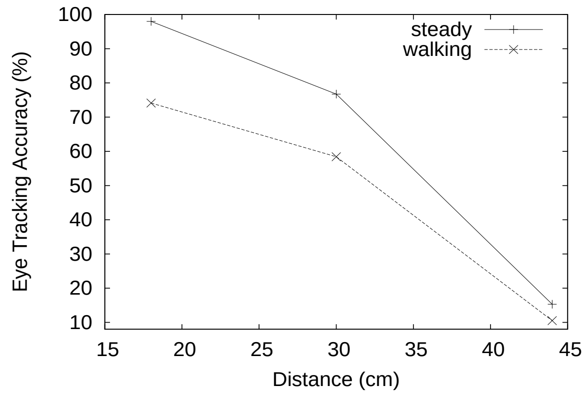

Figure 3: Eye tracking accuracy for the middle-center position as a function of different distances between the N810 and the eyes when the person is steady and walking.

[2]The reason we use the N810 instead of the Nokia N900 smartphone is because we obtain better quality video frames from the front camera. However, with its 600 MHz processor and up to 1 GB of application memory the N900 presents a more powerful hardware than the N810. Hence, we expect better performance (computation and inference delay) when running EyePhone on the N900. This is part of our on-going work.

Table 2: Average CPU usage, RAM usage, and computation time for one video frame. The front camera supports up to 15 frames per second. The last column reports the percentage of used battery by EyePhone after a three hour run of the system.

CPU	RAM	Computation time	Battery used after 3h
65.4%	56.51%	~100 msec	40%

are shown in Figure 3. As expected, the accuracy degrades for distances larger than 18-20 cm (which is the distance between the eye and the N810 we currently use during the eye template training phase). The accuracy drop becomes severe when the distance is made larger (e.g., ~45 cm). These results indicate that research is needed in order to design eye template training techniques which are robust against distance variations between the eyes and the phone.

System Measurements. In Table 2 we report the average CPU usage, RAM usage, battery consumption, and computation time of the EyePhone system when processing one video frame – the N810 camera is able to produce up to 15 frames per second. EyePhone is quite lightweight in terms of CPU and RAM needs. The computation takes 100 msec/frame, which is the delay between two consecutive inference results. In addition, the EyePhone runs only when the eye pair is detected implying that the phone resources are used only when a person is looking at the phone's display and remain free otherwise. The battery drain of the N810 when running the EyePhone continuously for three hours is shown in the 4th column of Table 2. Although this is not a realistic use case, since a person does not usually continuously interact with their phone for three continuous hours, the result indicates that the EyePhone algorithms need to be further optimized to extend the battery life as much as possible.

4.1 Applications

EyeMenu. An example of an EyePhone application is EyeMenu as shown in Figure 4. EyeMenu is a way to shortcut the access to some of the phone's functions. The set of applications in the menu can be customized by the user. The idea is the following: the position of a person's eye is mapped to one of the nine buttons. A button is highlighted when EyePhone detects the eye in the position mapped to the button. If a user blinks their eye, the application associated with the button is lunched. Driving the mobile phone user interface with the eyes can be used as a way to facilitate the interaction with mobile phones or in support of people with disabilities.

Car Driver Safety. EyePhone could also be used to detect drivers drowsiness and distraction in cars. While car manufactures are developing technology to improve drivers safety by detecting drowsiness and distraction using dedicated sensors and cameras [22], EyePhone could be readily usable for the same purpose even on low-end cars by just clipping the phone on the car dashboard.

5. FUTURE WORK

We are currently working on improving the creation of the open eye template and the filtering algorithm for wrong eye contours. The open eye template quality affects the accuracy of the eye tracking and blink detection algorithms. In particular, variations of lighting conditions or movement of the phone in a person's hand might make the one-time template inaccurate by not matching the current conditions of the user. A template created in a bright environment might work poorly to match the eye in a darker setting. Similarly, an eye template created when the person is stationary does not match the eye when the person is walking. We observe the implications of the one-time template strategy in the results presented in Section 4. It is important to modify the template generation policy in order for the system to be able to either evolve the template according to the encountered contexts if the template is generated in a context different from the current one, or, create new templates on the fly for each of the encountered settings (e.g., bright, dark, moving, etc.). In both cases the template routine should be fast to compute and minimize the resources used.

A second important issue that we are working on is a filtering algorithm that minimizes false positives, (i.e., false eye contours). One way to solve this problem is by using a learning approach instead of a fixed thresholding policy. With a learning strategy the system could adapt the filter over time according to the context it operates in. For example, a semi-supervised learning approach could be adopted, having the system evolve by itself according to a re-calibration process every time a completely new environment is encountered. In order to be sure the filter is evolving in the right direction the user could be brought into the loop by asking if the result of the inference is correct. If so, the new filter parameters are accepted, otherwise, discarded. Clearly, proper and effective user involvement policies are required, so prompting is not annoying to the user.

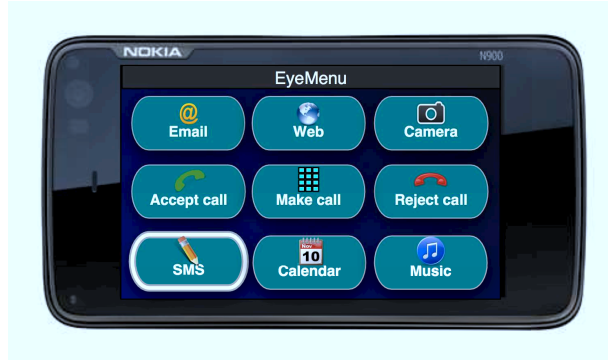

Figure 4: EyeMenu on the Nokia N900. While looking the display, a button is highlighted if it matches the eye position on the display. The highlighted button is ready to be "clicked" with a blink of the eye. In this example, the user is looking at the SMS button and the SMS keypad is launched by blinking the eye.

6. RELATED WORK

There are a number of developments in HCI related to mobile phones over the last several years [4, 5, 23, 6, 7, 8, 9, 10]. Some of this work exploits accelerometers on the phone in order to infer gestures. The PhonePoint Pen project [4] is the first example of a phone being transformed into a pen to write in the air in order to quickly take small notes without needing to either write on paper or type on a computer. A similar project evaluated on the iPhone has recently been proposed [8]. The uWave project [7] exploits a 3-D accelerometer on a Wii remote-based prototype for personalized gesture recognition. Phones can be used as radars as in [10] where proximity based sensing is used to infer speed of surrounding subjects.

Custom-built sensors interfaced with a mobile phone are used for driving the phone user interface by picking up eye movement [6] or converting muscular movement into speech [9]. The drawback of these technologies is that they require the support of external sensors attached to the mobile phone which might be an obstacle towards the large scale adoption of the technology.

Eye movement has recently been used for activity recognition [24]. By tracking the eye researchers show that it is possible to infer the actions of watching a video, browsing the web or reading notes.

The work in [5, 23, 11, 13] is more closely related to the EyePhone system. The eyeSight [5] technology exploits the phone camera to control the mobile phone by simple hand gestures over the camera. The openEye project [23] relies on an eyeglasses mounted camera to track a person's eyes to realize HCI applications. The Eye Mouse project [11] and [13] are designed for desktop machines with fixed cameras pointed at the user to pick up eye movement and use the eye as a mouse in [11], or to enable general HCI applications [13]. However, these systems are for static machines, using a specific fixed infrastructure of external sensors and cannot be easily replicated on mobile phones.

7. CONCLUSION

In this paper, we have focused on developing a HPI technology solely using one of the phone's growing number of onboard sensors, i.e., the front-facing camera. We presented the implementation and evaluation of the EyePhone prototype. The EyePhone relies on eye tracking and blink detection to drive a mobile phone user interface and activate different applications or functions on the phone. Although preliminary, our results indicate that EyePhone is a promising approach to driving mobile applications in a hand-free manner. A video of the EyePhone demo can be found at [15].

8. ACKNOWLEDGMENTS

This work is supported in part by Nokia, Intel Corp., Microsoft Research, NSF NCS-0631289, and the Institute for Security Technology Studies (ISTS) at Dartmouth College. ISTS support is provided by the U.S. Department of Homeland Security under award 2006-CS-001-000001, and by award 60NANB6D6130 from the U.S. Department of Commerce. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any funding body. A special thanks goes to Prof. Lorenzo Torresani for his valuable input on this work.

9. REFERENCES

[1] Apple iPhone. http://www.apple.com/iphone.
[2] Nokia Series. http://europe.nokia.com/nseries/.
[3] HTC. http://www.htc.com/www/product.
[4] S. Agrawal, I. Constandache, S. Gaonkar, and R.R. Choudhury. PhonePoint Pen: Using Mobile Phones to Write in Air. In *MobiHeld '09*, pages 1–6. ACM, 2009.
[5] eyeSight. http://www.eyesight-tech.com/.
[6] DoCoMo eye control. http://tinyurl.com/ycza2jd.
[7] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: Accelerometer-Based Personalized Gesture Recognition and its Applications. In *Pervasive and Mobile Computing*, vol. 5, issue 6, pp. 657-675, December 2009.
[8] Accelerometer-Based Gesture Recognition with the iPhone. http://tinyurl.com/yjddr8q.
[9] The Mobile Phone that Could "Read Lips". http://www.cellular-news.com/story/42211.php.
[10] Nokia Mobile Radar. http://tinyurl.com/yfo2dtd.
[11] The Eye Mouse project. http://www.arts.ac.uk/research/eyemouse/index.htm.
[12] F. Karray, M. Alemzadeh, J.A. Saleh, and M.N. Arab. Human-Computer Interaction: Overview on State of the Art. In *International Journal on Smart Sensing and Intelligent Systems*, Vol. 1, No. 1, March 2008.
[13] M. Chau and M. Betke. Real Time Eye Tracking and Blink Detection with USB Cameras. In *Boston University Computer Science Technical Report No. 2005-12*, 2005.
[14] E. Miluzzo, C.T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, A.T. Campbell. Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones. In *Eighth International ACM Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, San Francisco, CA, USA, June 15-18, 2010.
[15] EyePhone demo video. http://sensorlab.cs.dartmouth.edu/eyephone/demo.m4v.
[16] Nokia n900. http://maemo.nokia.com/n900.
[17] Eye Tracking and Blink Detection Library. http://tinyurl.com/yb9v4f2.
[18] JP Lewis. Fast Normalized Cross-Correlation. In *Vision Interface*, volume 10, pages 120–123. Citeseer, 1995.
[19] Nokia n810 tablet. http://www.nokiausa.com/find-products/phones/nokia-n810.
[20] OpenCV. http://opencv.willowgarage.com/wiki.
[21] GStreamer. http://www.gstreamer.net/.
[22] Mercedes-Benz Drowsiness Detection System. http://tinyurl.com/353u8h6.
[23] D. Li, J. Babcock, and D.J. Parkhurst. openEyes: a Low-Cost Head-Mounted Eye-Tracking Solution. In *2006 Symposium on Eye Tracking Research & Applications*, page 100. ACM, 2006.
[24] A. Bulling, J.A. Ward, H. Gellersen, and G. Troster. Eye Movement Analysis for Activity Recognition. In *11th International Conference on Ubiquitous Computing*, pages 41–50. ACM, 2009.