GlimpseData: Towards Continuous Vision-Based Personal Analytics

Seungyeop Han
University of Washington

Rajalakshmi Nandakumar
University of Washington

Matthai Philipose
Microsoft

Arvind Krishnamurthy
University of Washington

David Wetherall
University of Washington

ABSTRACT

Emerging wearable devices provide a new opportunity for mobile context-aware applications to use continuous audio-/video sensing data as primitive inputs. Due to the high-datarate and compute-intensive nature of the inputs, it is important to design frameworks and applications to be efficient. We present the *GlimpseData* framework to collect and analyze data for studying continuous high-datarate mobile perception. As a case study, we show that we can use low-powered sensors as a *filter* to avoid sensing and processing video for face detection. Our relatively simple mechanism avoids processing roughly 60% of video frames while missing only 10% of frames with faces in them.

Keywords

Continuous mobile perception; Mobile phone sensors; Face detection; Filtering

Categories and Subject Descriptors

I.4.8 [**Scene Analysis**]: Sensor fusion

1. INTRODUCTION

The past few years have seen the emergence of *personal analytics*, also dubbed *the quantified self movement* [1], where people use sensors to measure, analyze and share data about their lives, as a significant business and societal trend. Although the metric they quantify varies, products to date have focused on low datarate sensors such as inertial sensors, location, physiological sensors (e.g., weight and heart rate), environmental sensors such as motion sensors and even virtual feeds such as Twitter. Precisely because they manipulate little data, such systems have the virtue of requiring little power from sensors and wearable devices and of reducing privacy costs to their users. Relative to systems based on high-datarate sensors such as vision and audio, however, these systems sacrifice much potentially valuable insight about their users. In this paper, we take an early step toward analyzing video and audio streams in addition on a continuous basis to derive personal analytics.

The notion that visual data from wearable devices can provide a rich complement to other sensor streams is not new. Work from at least the early nineties has demonstrated the utility of vision to translate American Sign Language [20], play indoor virtual-reality based games [19], recognize the facial expression of wearers [15], and mine life patterns at various scales [4]. Although these efforts have contributed much to highlight the potential of wearable vision to understand wearer context, their visual analysis components were fairly brittle and restricted to small numbers of contexts. A parallel line of research, in computer vision, has arrived at a point where faces [2], objects [10, 16] and locations [9] in video streams can be recognized with increasing accuracy in very large numbers via the use of large-scale machine learning techniques. Merging these lines of work, we see an opportunity in wearable systems that, over the roughly 10 hours a day they are worn, *continuously* recognize, analyze and act on every person, place, object and activity experienced by their user.

Performing mobile vision-based analysis *at scale* in this manner poses two major related systems challenges. First, how should modern vision algorithms be complemented with other sensors, accelerated using today's heterogeneous processing resources (such as GPUs, DSPs and FPGAs) and constrained by use cases such that their recognition rates in the wearable setting move from the "interesting" level to "useful"? Second how can the lower datarate sensors be used to minimize the use of the higher datarate sensors such that the latter are only used when they are likely to provide information not available to the former? More broadly, given that today's computer vision algorithms applied to video consume well over 100x the power budget of today's phones [6], is it possible to build a wearable device that performs vision (or visual data handoffs) on a continuous basis on the 200mAh power budget realistic for a wearable device in the foreseeable future?

In this paper, we take two early steps toward answering these questions. First, we describe a data collection and analysis framework we have implemented (and intend to release publicly) on the popular Android platform to help collect continuous sensor-augmented visual data from daily

WPA'14, June 16, 2014, Bretton Woods, New Hampshire, USA.
Copyright 2014 ACM 978-1-4503-2825-8/14/06 ...$15.00.
http://dx.doi.org/10.1145/2611264.2611269.

life. The Android app and data visualizer can be found at: `https://github.com/UWNetworksLab/GlimpseData`. Although the architecture of the framework is similar to those released for lower datarate sensors [11] (except that handling video poses synchronization, power and storage challenges), we hope that having a convenient platform will help kick-start work in this area. Further, we note the severe privacy challenges of this kind of data collection and hope to discuss best practices and the development of a common, privacy-vetted research dataset with workshop participants.

Our second contribution is a technical case study building on the data we capture from the system above. We focus on the problem of predicting whether a given video frame will contain faces of conversation partners in it *using only low-datarate sensors*. We believe that *filter* stages of this kind that have very high recall and modest precision in recognition, while consuming relatively little power, will be an important aspect of developing pipelines that have low overall cost. We demonstrate that indeed, standard learning techniques can yield a filter that can discard 60% of all frames using no visual analysis while preserving for further analysis over 90% of frames that *do* have faces in them.

We consider these to be very early steps in addressing the challenge of continuous vision-based perception in a mobile setting. We therefore discuss possible future steps in this direction.

2. DESIGN AND IMPLEMENTATION

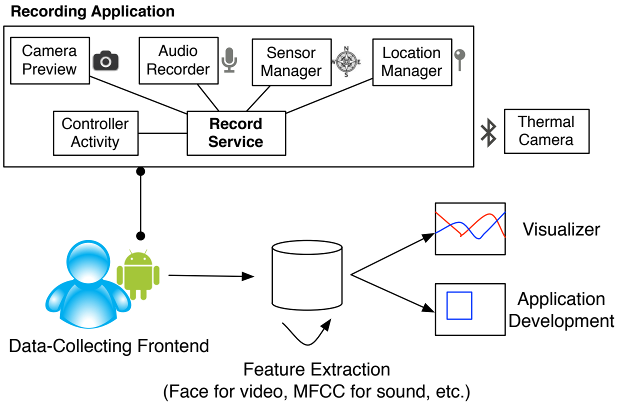

Figure 1: Overview of the workflow

In this section, we present the design and implementation of the GlimpseData framework for collecting and analyzing vision-augmented sensor streams on the phone. Figure 1 shows the overview of the workflow. It consists of a data-collecting front-end as an Android application, data-processing tools, and a visualization tool.

2.1 Data-Collecting Front-end

The very first step to study mobile perception is to collect all the possible context information surrounding users. While wearable video accessories are not yet widely available, we can use smartphones as an proxy to collect data. Modern smartphones allow applications to access a wide range of contextual data. For example, applications can get video stream from a camera, and retrieve real-time values from low-datarate sensors like accelerometers. We build an Android application so that researchers can use smartphones as a front-end for data collection. As shown in Figure 1, the recording application runs an Android service that registers with Android components to get updates when a user initiates recording. Running as a service allows it to run in the background without a screen turned on and even during the use of other applications. The data-collecting application currently gathers the following data:

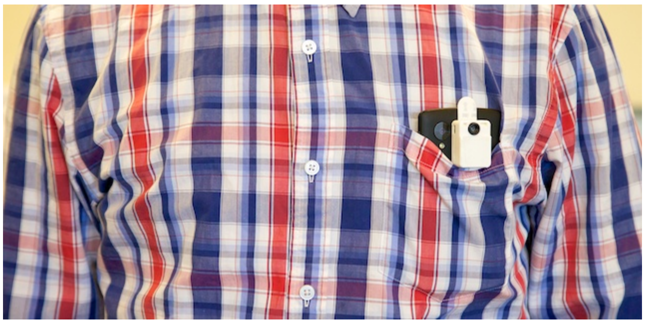

Figure 2: A person wearing a smartphone to collect data.

Video: It records video via a built-in camera. As shown in Figure 2, a user may put a phone into his pocket while collecting data, and use the rear-camera to capture frames. While the angle taken by this setting is different from a wearable device like Google Glass, it can provide a good approximation of it. A naïve way to collect video frames is to use `MediaRecorder` provided by Android, however, it is not possible to get accurate timestamp of the recorded frames with MediaRecorder due to inconsistent delay before recording starts. Thus, we instead use preview frames (at roughly 5 fps) obtained from `onPreviewFrame` function.

Audio: As our application collects video frames by using preview screen, audio data needs to be recorded separately. Audio is recorded through MediaRecorder into `.3gp` format, which is later converted into `.wav` for analysis. Audio recording with MediaRecorder incurs consistently small delay (<100ms) before recording starts unlike video recording.

Sensor Data: Modern smartphones have various built-in sensors to measure motions and environmental conditions. The application registers to receive updates for every sensor in the device, including the accelerometer, gyroscope, light, proximity, compass, gravity and if available, pressure and ambient temperature.

Location: It monitors GPS, and network-based location. In addition, it logs SSID of WI-FI APs by periodic scan.

Thermal Camera: In addition to above data available from any Android phones, we attach a custom-built thermal camera close to the phone's RGB camera (the small white box shown in Figure 2). It continuously reports the average *temperature* over the field of view (FOV) of each of its pixels as a 16x4 float array over a 40x15° total FOV. The application communicates with the thermal camera via bluetooth and logs it with timestamp.

2.2 Visualizer

Visualization of collected data helps in a sanity check and understanding it. Our data is a combined set of video frames

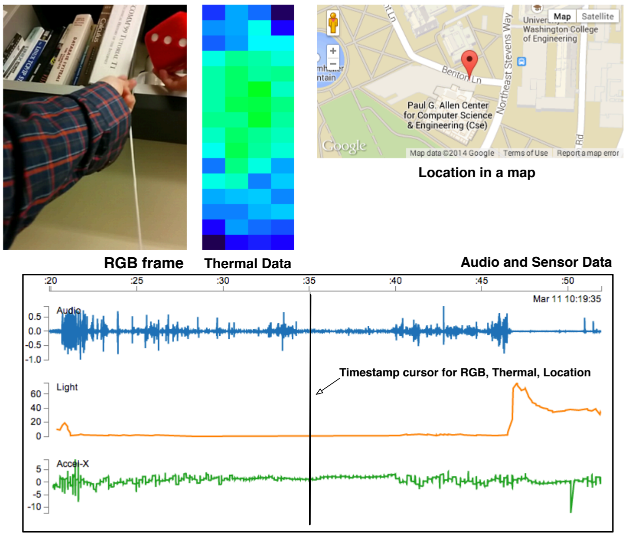

Figure 3: A screenshot from the visualizer

with timestamp, time-series data of various sensor values, and audio data with starting timestamp. We build a simple data visualizer which allows users to navigate through timeline. It is written in javascript based on `D3.js`[1]. After few pre-processing steps to create a list of frames and to convert audio files, users can access the data through any web browser. Figure 3 shows a screen capture from the visualizer showing an example collected through the data-collection application.

The view of the visualizer comprises two large panes. One area at the top of the screen presents a RGB frame, thermal data, and a map pointing the location where the frame is recorded. Thermal data is shown as a heatmap. For example, the area where a hand is in the example figure is painted as green, which has higher temperature than the surrounding blue/black areas. The bottom area is timeseries graphs for audio and sensor values. Moving a cursor on the timeseries updates all data displayed appropriately.

3. DATASET

total duration	116 mins
days	7
video frames	33,769
video frames w/ detected faces	1,664 (4.93%)
thermal camera frames	100,347
accelerometer readings	1,046,285
gyroscope readings	1,045,817

Table 1: Summary of the dataset

By using the Android application described in the previous section, one of the authors collected a dataset on an LG Nexus 4 running Android 4.4. Table 1 summarizes some statistics of the dataset . The dataset is sampled from 7 days of daily life, such as walking in different places, riding a bus, driving a car, working on a desk, and talking to others. Although this dataset is by no means comprehensive, we believe it illustrates many essential characteristics

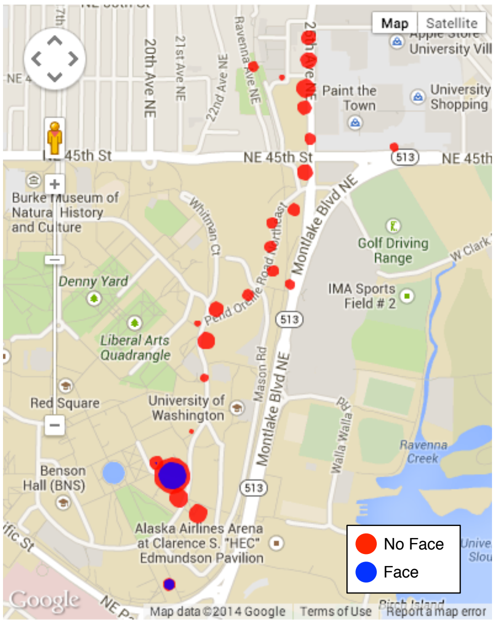

Figure 4: Locations with and without faces. Circle size is proportional to log of the number of frames collected at the location. Nearby frames within 50m are clustered into one circle.

of personal video footage aimed at personal analytics. We use this data to drive the case study of the next section.

In what follows, we will focus on predicting, using sensors other than video, whether faces appear in each dataset frame. We used the HaarCascade face detector from `OpenCV`, with manual verification of detected faces to avoid false positives, as baseline. Out of 33,769 frames in the dataset, 4.93% frames contain at least a face. Figure 4 shows a part of a map with locations where the faces are detected in red and the remaining locations in blue.

4. CASE STUDY: FILTERING FOR FACE DETECTION

Face detection and recognition can be used as important primitives for various applications. For example, in speaker identification [12], the application may use recognized faces, in addition to audio data, to find out whom the user talked to during a day

In conventional face detection with a wearable video device, it is every frame of the streamed video must be processed to provide the context information to applications. However, having a face in a frame is a rare event as shown in our dataset from the previous section. The tasks of sensing a RGB frame and running a face detection algorithm are expensive in terms of battery consumption and are often unnecessary.

In this study, we ask the following question: *"by using low-power sensors, can we determine if a frame is unlikely to have a face before running a face detector over a RGB frame?"* Figure 6 illustrates how the system works. Intuitively, there are sensors that can be used for filtering the non-face frames: if the frame is too dark, the face detector will not find a face; if the user is moving too quickly—calculated by accelerometer and gyroscope—frames can be

[1] http://d3js.org/

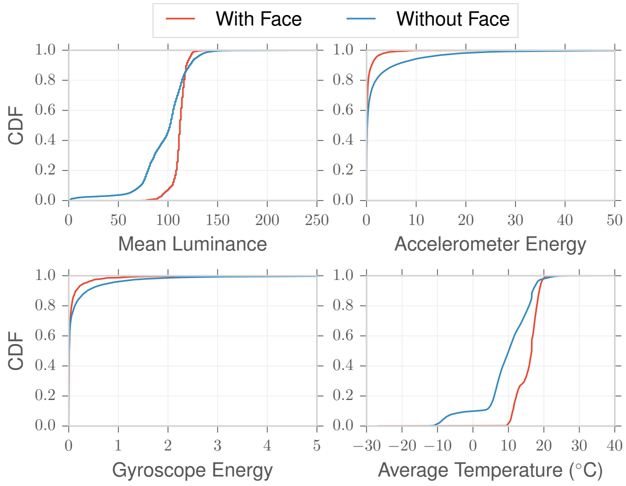

Figure 5: Correlations between a sensor and appearance of face in a frame shown by CDFs of the sensor values.

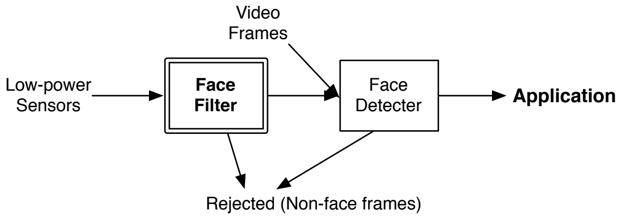

Figure 6: Illustration of how filtering works

too blurred to detect faces. More notably, the thermal camera can be a good indicator, because the temperature of a human body is typically higher than that of the surroundings. These sensors consume relatively little power (accelerometer and gyroscope: 10 mW, light sensor: 3 mW, thermal camera: 9 mW, GPS: 140 mW at under low duty cycle), compared to the RGB imager (over 200 mW for reading, and over 600mW for analysis). These are representative numbers for the phone we currently use; with state-of-the art variants of sensors, we expect the relative gap between video analysis and low-datarate sensors to be even higher.

For the filter to be efficient, it needs to filter out as many frames as possible while not missing frames with faces. So the evaluation metric here is the fraction of frames to process compared to the potential number of frames without any filtering, and recall for the frames with faces; ideal case should be when the proportion of frames to process is close to 0 and recall is close to 1.

First, we look into correlations between each sensor and likelihood of having faces to check whether it would be effective to use the measurements from the low power sensors as a filtering mechanism. Note that we could not use light sensor data since a light sensor in the phone is located in front of the phone and it faced into the pocket during the collection time. Instead, we estimate brightness by calculating average luminance value of each frame from a YUV colorspace and use it hereafter. Figure 5 shows the distributions of the values from each sensor for frames with and without faces. As expected, sensor values with faces and without faces have distinct distributions, which means that each sensor can indicate likelihood of faces at some extent. However, as shown in the graphs, values for those classes have a large overlap as well; simple thresholding for filtering non-face frames with one sensor may not work. Figure 7 depicts the proportion of frames to process and recall of frames with faces when applying thresholds on each sensor value. While thermal camera and light value show better performance in filtering out uninteresting frames than the other two sensors, they miss many frames with faces as well. To filter out more than 60% frames, it misses more than 25% of frames having faces.

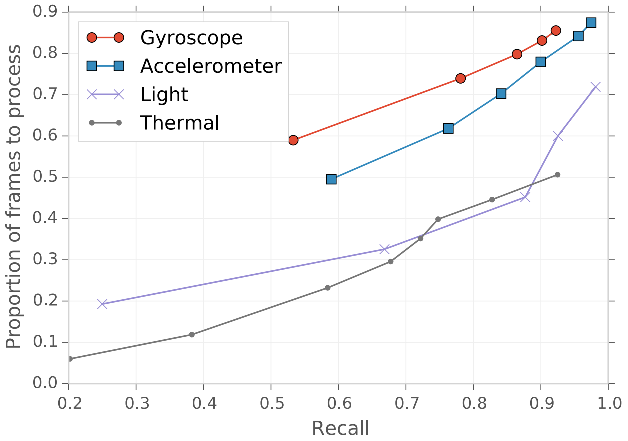

Figure 7: Fraction to process and recall curve when applying thresholds on each sensor's values. (Lower line is better)

input sensor	features	# features
accelerometer[2]	energy $(x^2 + y^2 + z^2)$	1
gyroscope	energy $(x^2 + y^2 + z^2)$	1
light	mean luminance	1
thermal	raw value, avg, max	66
sound	MFCC over 25ms	13
location	latitude, longitutde	2
Overall		83

Table 2: Features used in the logistic regression

To consider all those possible factors into the calculation, we build a logistic regressor. For each frame, it finds closest sensor values and uses them as features in training and testing phases. Table 2 describes which features are used in the logistic regression. Instead of classifying a frame as with face if probability from the regressor is higher than 0.5, we vary the probability threshold. In fact, it worked well with small probability threshold lower than 0.1. Figure 8 shows the result of logistic regression. For each run, we used randomly picked 90% of frames as a training set and the rest as a testing set, and repeated 10 times. It filters out 70% of frames while missing only 10% of frames with faces, which is much better than applying a threshold to one sensor value.

The above is a best-case since, by selecting training and test sets randomly we likely (over)fit to neighboring time steps: given a face in a training frame, there will more likely be a face in any adjacent test frame. To verify generaliza-

[2] We used linear accelerometer data which excludes gravity from accelerometer data.

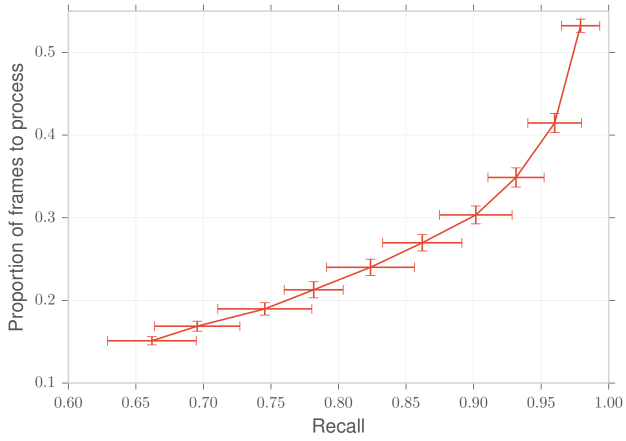

Figure 8: Logistic regression result with varying the probability threshold (error bars are 1 σ).

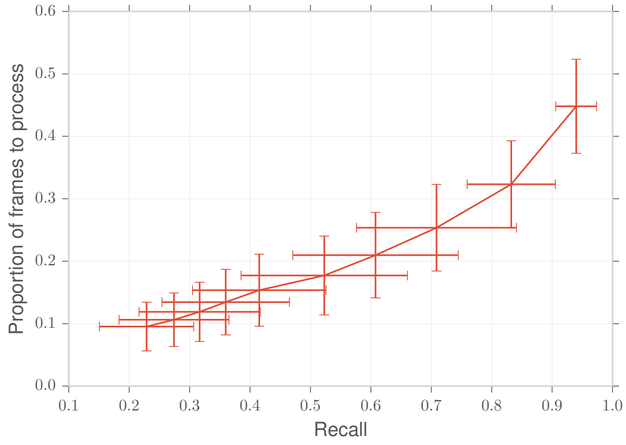

Figure 9: Logistic regression result with clustering 1000 frames for training and testing.

tion at least across a few minutes, we segment the data into 1000-frame windows (about 3∼4 minutes each), pick 90% of clusters as a training set, and the rest as a testing set. Note that there is little correlation in the presence of faces across frames separated by minutes. To avoid imbalance of positive samples, we exclude folds where the randomly chosen testing split has less than 5%, or more than 20%, of all face frames. Figure 9 shows the result. While performance is somewhat worse, the system still filters 60% of frames while falsely rejecting 10% of frames with faces.

5. DISCUSSION

While building the data collection application, we noticed a couple of issues that need to be addressed for improving the quality of dataset. First of all, more accurate timestamping is required, especially for real-time reasoning applications. As it gathers data from multiple sources, synchronizing data points is not trivial. We improved timestamp accuracy by using preview frames instead of media recorder for video. However, it still has a consistent delay of 2-3 frames. In addition, currently the number of frames per second is smaller (5-6fps) than the target number (30fps). Improving this will be helpful in gathering better dataset.

For the collected dataset to be widely useful, having a framework to support sharing data with research community. However, the dataset contains much private information. Integrating privacy-preserving solutions as in GigaSight [18] can be helpful, while some applications may not work after applying it. More broadly, it is important for the research community to adopt a set of standards for data collection, and given the delicateness of collection, collaborate in the collection of a single high-quality corpus.

The case study we presented is a starting step toward a system design for continuous mobile perception. There is much more to be done. For example, filtering needs to be generalized so that it works with multiple applications, which requires the ability to specify requirements per application and to coalesce filters. Further, it should be possible to automatically and optimally pick sensors for individual stages of a multi-stage filter. The theory of cost-sensitive classifiers [22] is relevant. Once promising windows are detected, classifying these windows over many classes (e.g., faces, places or objects) will likely need innovations in compute elements, system design and classification algorithms.

6. RELATED WORK

We discussed related work in the wearable and vision communities in the introduction. Here we restrict ourselves to work in the systems community.

There have been a number of studies on continuous sensing in mobile applications. SenseCam captures pictures every 30 seconds along with other sensor data, which is designed for retrospective memory aid [7]. The dataset and visualization are similar to ours, but its low frame rate is not suited to the real-time applications that we target. Audio data is also widely used in detecting sound events [13] and speaker identification [12]. Another application is indoor localization where sensors like accelerometers and magnetometers are used to track continuously in indoor spaces [3, 21].

Prior research has also targeted energy efficiency of continuous sensing. Jigsaw [14] uses pipelining and conditionally triggers high-energy stages to reduce energy costs. Seemon [8] optimizes for processing context monitoring queries by choosing an essential sensor set for the queries. None of these systems target video.

Moving to video data, GigaSight proposes a scalable method to crowdsource mobile video content by decentralizing the cloud architecture using VM-based cloudlets [18]. The focus is data collection, not power-aware visual analysis. Finally, recent work [5, 17] advocates migrating workload into the cloud. These approaches are complementary to using sensors to filter frames: filtering is likely essential for offloading at a reasonable power budget.

7. CONCLUSION

We have presented GlimpseData, a framework to collect and analyze data for studying continuous high-datarate mobile perception. It includes an Android application that collects data from multiple sources including video and sensors, and a web-based visualizer that allows researchers to navigate collected data easily. With data collected through the application, we presented a case study of filtering unnecessary frames for face detection with low-powered sensors. Our early results are promising: we can filter out 60% of frames without processing visual inputs while missing only 10% frames of interests. Although clearly an enormous challenge, we believe that adding vision to personal analytics could be truly revolutionary.

8. ACKNOWLEDGEMENTS

We thank Steve Hodges and Stuart Taylor of Microsoft Research Cambridge for providing us with the prototype infrared camera used in this study. This material is based on research sponsored by DARPA under agreement number FA8750-12-2-0107. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

9. REFERENCES

[1] Quantified self website. `http://quantifiedself.com/`.

[2] Deepface: Closing the gap to human-level performance in face verification. 2014.

[3] K. Chintalapudi, A. P. Iyer, and V. Padmanabhan. Indoor Localization without the pain. In *Mobicom*, 2010.

[4] B. Clarkson. *Life Patterns: structure from wearable sensors*. PhD thesis, MIT, 2002.

[5] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: Making smartphones last longer with code offload. In *MobiSys*, 2010.

[6] S. Han and M. Philipose. The case for onloading continuous high-datarate perception to the phone. In *HotOS*, 2013.

[7] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood. SenseCam: A Retrospective Memory Aid. In *Ubicomp*, 2006.

[8] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. SeeMon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *Mobisys*, 2008.

[9] K. Konolige, J. Bowman, et al. View-based maps. In *Proceedings of Robotics: Science and Systems*, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[11] N. D. Lane, E. Miluzzo, et al. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, 2010.

[12] H. Lu, A. J. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu. SpeakerSense: Energy Efficient Unobtrusive Speaker Identification on Mobile Phones. In *Pervasive*, 2011.

[13] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. SoundSense: Scalable sound sensing for people-centric applications on mobile phones. In *Mobisys*, 2009.

[14] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The Jigsaw continuous sensing engine for mobile phone applications. In *SenSys*, 2010.

[15] R. W. Picard and J. Healey. Affective wearables. In *ISWC*, 1997.

[16] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012.

[17] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan. Odessa: Enabling interactive perception applications on mobile devices. In *Mobisys*, 2011.

[18] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. Scalable crowd-sourcing of video from mobile devices. In *Mobisys*, 2013.

[19] T. Starner, B. Schiele, and A. Pentland. Visual contextual awareness in wearable computing. In *ISWC*, 1998.

[20] T. Starner, J. Weaver, and A. Pentland. A wearable computer based american sign language recognizer. In *ISWC*, 1997.

[21] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-Drive: Unsupervised Indoor Localization. In *Mobisys*, 2012.

[22] Q. Yang, C. Ling, X. Chai, and R. Pan. Test-cost sensitive classification on data with missing values. *Knowledge and Data Engineering, IEEE Transactions on*, 18(5):626–638, 2006.