

**Wearable-Assisted Social Interaction as Assistive
Technology for the Blind**

by

David S. Hayden

Submitted to the Department of Electrical Engineering and Computer
Science

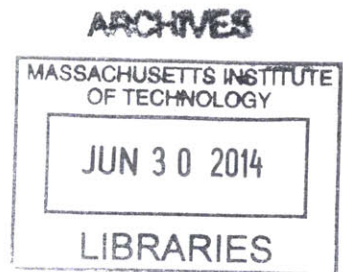
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014



© Massachusetts Institute of Technology 2014. All rights reserved.

Signature redacted

Author
Department of Electrical Engineering and Computer Science
February 14, 2014

Signature redacted

Certified by
Seth Teller
Professor
Thesis Supervisor

Signature redacted

Accepted by
Leslie A. Kolodziejki
Chairman, Department Committee on Graduate Theses

**Wearable-Assisted Social Interaction as Assistive Technology for the
Blind**

by

David S. Hayden

Submitted to the Department of Electrical Engineering and Computer Science
on February 14, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

This work presents an end-to-end wearable system designed to learn and assist its (potentially blind) wearers with daily social interactions. In particular, it visually identifies nearby acquaintances and provides timely, discreet notifications of their presence to the wearer. Offline, the system learns the people with whom the wearer interacts by automatically detecting social interactions through egocentric audio, video and accelerometer data and querying the wearer for the identities of persons unknown to the system.

Thesis Supervisor: Seth Teller

Title: Professor

Acknowledgments

This work wouldn't have been possible were it not for the examples set by family and loved ones: from my mother came the drive to succeed; my father, the vision to dream; my brother, the realization of desired outcomes. From Don came a modicum of so-called common sense (though it seems that will forever be a work-in-progress). From Colleen, joy of my life, came a fuller understanding that my work matters, that decisions made now shape the foundations we jointly build for the future. To each, I say that the love and support expressed through your encouragement, patience and understanding has always helped me feel firmly grounded and thus able to push ever onward. Thank you. I hope that I am always able to reciprocate such warmth.

To my adviser, Seth: working with you has and continues to be a pleasure, and instrumental to my growth as an academic and a professional. Though much remains, and always will, I have a better understanding and certitude of what that entails, and am becoming ever more independent, even as I reach out to and work with others. Thank you for your support and exhortations as well as your patience and confidence.

Many others have played a role in enabling this work, but I should like to point out several in particular. I thank Sophie Diehl for her excellent tailoring (whose contributions can be seen in the appendix). I thank Ross for pressing me to get started on the writing. And I thank other mentors, past and present, including John, Terri, Panch, Glenn, David and especially Steve for the roles they've played in my professional development. I hope I can pay forward all the timely guidance each of you has provided.

Contents

1	Introduction	15
1.1	Social Interaction for the Blind	16
1.2	Wearable Computing and Social Interaction	18
2	Wearable System	21
2.1	Overview	21
2.2	Design Considerations	23
2.3	Physical Design	27
2.4	Hardware	28
2.5	System Architecture	31
2.5.1	Network Connections and Data Transport	34
2.6	User I/O	37
2.6.1	Vibration-Encoded Messages	38
2.7	Detection and Recognition	41
2.8	Logging	42
2.9	User Labeling of New Acquaintances	42
3	Detecting Social Interactions	43
3.1	AUC-Boosted Person Detection	43
3.1.1	Introduction	43
3.1.2	Features	45
3.1.3	Data	45
3.1.4	Weak Classification	46

3.1.5	AUC Boosting and Cascade	46
3.1.6	Results	49
3.1.7	Limitations	51
3.1.8	Summary	53
3.2	Social Interaction Detection	53
4	Sightless Image Labeling	57
4.1	Appearance-Based Clustering	58
4.1.1	Adjusted Mutual Information	59
4.2	Label Correspondence and Sub-Sample Selection	60
4.3	Labeling Interface	61
5	Conclusions	65
5.1	Utility, Privacy and Physical Comfort	65
5.2	Future Work	66
A	Figures	69

List of Figures

2-1	Workflows for our wearable system. Boxes in white are models that we build and train prior to system use; they are updated infrequently. The green workflow is performed in real-time whereas the red workflow is performed offline, at the end of each day of use.	24
2-2	Human models of wearable camera angle of view (i) and wearable camera motion (ii). In Figure (i), angle of view is computed by casting rays 37.5mm from each polygonal face of the body in all directions and rejecting those that are obscured by other parts of the body. While all placements will be obscured by the body behind them, some will be more obscured in outward views than others. In Figure (ii), (a) shows motion at each step of a human walk while (b) shows the cumulative motion that a wearable camera is subjected to throughout a complete walk. Darker regions are more desirable and mean greater field of view (i) or less motion (ii). Figures adapted from [24].	25
2-3	How does the public feel about wearable cameras? We asked 100 people to rate the social acceptability of different wearable configurations worn by four people. Our results (bottom) suggest that the best wearable camera may be the one that is not seen.	27
2-4	Our wearable system in two configurations. It is most discreet in the Jacket Configuration (a), but most easily donned and worn by different people in the Lanyard Configuration (b).	28

2-5	Watch used in our wearable system for user input (with the three buttons on the right) and output (via glanceable text display or vibration-encoded text).	30
2-6	Optional accessories for our wearable system. Left: a Bluetooth headset for personal audio. Right: a Bluetooth remote placed within a pocket and used for discreet cueing. All buttons are mapped to a common function so that the wearer need only press against the outside of their pocket.	31
2-7	Software architecture of our wearable system. The various Handlers (left) are available to each user-visible Mode, which itself inherits from a Videation-Mode interface. This design makes it easy to implement new or change existing Modes.	33
2-8	Definitions for testing object serialization in the LCM, Protocol Buffer and MsgPack Object Serialization libraries.	35
2-9	Serialization definition for the canonical VideationMessage, which gets passed between the wearable system and server, with fields filled in as appropriate. Note that the only required field is the timestamp, and that only the fields necessary for a given task are transmitted (per the optional specifier).	36
2-10	Frequency-weighted character encoding time with expert and record human Morse-decoding performances.	39
2-11	Word encoding time (for PARIS) with expert and record human Morse-decoding performances.	40
3-1	The AUC-Boosted Cascade Algorithm. Note that we use $H_i(x)$ to denote the classifier resulting from training cascades $1 \dots i$, while $H_i^t(x)$ represents a classifier in the t^{th} round of boosting construction within a cascade i	48
3-2	ROC Curve of our final face detector; $AUC = 0.998$; in particular, we get 0.7 TPR on the test set with only $1e - 3$ FPR. Accepting slightly more false-positives quickly drives the TPR towards 1.	50

3-3	Training Error by boosting iteration. We note that the error is already extremely low after the first iteration suggesting that, indeed, our features are indeed discriminative	51
3-4	ROC Curve of a non-boosted, non-tuned HOG-based classifier; AUC is a weak 0.59 suggesting that further tuning and boosting would likely improve performance.	52
3-5	ROC Curves of SVM-based detection of social interactions using different groups of features. No single feature grouping (face, motion, audio) performs as well as their combination, which yields an AUC of 0.861. . . .	54
4-1	Images of the blind-accessible labeling interface, in which each social interaction is summarized, and the wearer is able to provide labels for the people they interacted with by listening to audio clips.	63
A-1	Materials and pattern specifications for the Jacket component of our wearable system, in which a wearable camera is discreetly placed within a pocket sewn into its inner-lining, with its lens peering out through a small hole.	70
A-2	Steps for creating the Jacket component of our wearable system in which a wearable camera is discreetly placed within a pocket sewn into its inner-lining, with its lens peering out through a small hole.	71
A-3	Images of people using different configurations of wearable cameras, shown to the public, and asked whether they felt the person in each image would look ‘normal,’ ‘peculiar,’ or ‘weird’ if they encountered them in a super market while shopping for groceries. In order are the control (no camera), ear camera, chest camera, pinwheel (for comparison), head camera and face camera.	72

List of Tables

2.1	Data serialization sizes and encode/decode times for streaming VGA-resolution JPEG images along with a timestamp in three modern serialization libraries. Data is encoded on a dual-core Android platform and sent over the WebSockets protocol to a quad-core Linux server.	35
4.1	Confusion matrix resulting from the use of appearance-based clustering to distinguish between two participants in a social interaction. Columns are predictions, rows are ground truth.	58
4.2	Confusion matrix resulting from the use of appearance-based clustering to distinguish between three participants in a social interaction. Columns are predictions, rows are ground truth.	59

Chapter 1

Introduction

This work presents an end-to-end wearable system designed to learn and assist with its (potentially blind) wearers daily social interactions. In particular, it visually identifies nearby acquaintances and provides timely, discreet notifications of their presence to the wearer. Offline, the system learns the people with whom the wearer interacts by automatically detecting social interactions through egocentric audio, video and accelerometer data and querying the wearer for the identities of previously unknown persons.

Particular to our work is the motivation to provide individuals who are blind or legally blind with the ability to more independently engage in social interaction. This application imposes constraints on how the user and system interact as, for example, a blind wearer cannot visually identify new acquaintances to her wearable through conventional means.

While work on wearable computing is becoming increasingly common, much of it focuses on passively modeling/predicting physical activities that the wearer engages in, e.g. energy expenditure in exercise, whether a wearer has fallen, and, more recently, on gaze/attention/role of social interaction. Little has been done in the way of using wearable computers to actively and unilaterally (i.e. for just one party) augment social interaction. Doing so requires unusual attention to the aesthetics of a wearable system lest the device do more harm than help to the wearer's social interactions. Thus, softer considerations including physical appearance, privacy, and methods of communication are considered throughout this work.

We motivate the practical utility and academic interest of a system that learns its wearers

social interactions from the perspectives of assistive technology and wearable computing.

Following, Chapter 2 details the system architecture, including how the system runs online and how its models are built or updated offline. This chapter includes motivation of and discussion on hardware, design, and interface considerations, and discusses how vision, audio and motion data are brought together for determining likelihood of and participation in social interactions.

Chapter 3 discusses the training of models necessary for our wearable system, acquaintance detection and social interaction detection in particular. Results on their performance and future needs for improvement are discussed.

Subsequently, Chapter 4 details our approach to a blind-accessible labeling framework in which a (potentially blind) wearer can teach their wearable system to assist with social interactions. Rather than ask wearers to visually label training images as might be done with sighted users they are instead asked to label small audio snippets localized in time to the people they interact with. Taken together, with the results of Chapters 2 and 3, this describes our end-to-end wearable system that can identify to and learn from its (potentially blind) wearer the acquaintances s/he interacts with.

Chapter 5 concludes with discussion of future work, practical limitations of the current system, and physical/privacy concerns for using wearable systems that attempt to augment everyday living, in this case, with respect to assistance wearers in social interaction.

1.1 Social Interaction for the Blind

Considerable research has been conducted on the socialization of individuals who are blind and visually-impaired. While there is consensus that they have fewer friends and engage in fewer social interactions [16], much of the work appears appears to focus on deficiencies in their ability to pick up on social cues, form accurate impressions of others, empathize or express themselves ('bland look, fixed smile, limited gesturing) as well having a tendency to exhibit maladaptive social behaviors, including inappropriate rocking, voice projection, and turning away from the speaker [18]. Other works additionally consider the correlation between physical attractiveness and the number and frequency of healthy friendships,

and suggest this contributes to the social deficiencies in BVI due to commonly associated aberrations in eye appearance or movement [32].

One issue that appears to have received little or no specific consideration is the reduced ability in the blind and visually-impaired to initiate social interactions. Limited or no vision reduces their ability to identify others and so it is reasonable to expect that they suffer from many of the same challenges that Prosopagnosics (people with ‘face blindness’) do [39]. The analogy is particularly apt for individuals with limited vision, who may not show obvious indicators of their impairment (e.g. use of a cane or seeing-eye dog). Challenges that they inherit, then, include reduced ability or tendency in initiating social interactions and inability to relate naturally to people that engage them in social interactions on account of the low-vision person being unable to recognize them.

Furthermore, among the approximately 15 million blind and legally blind working-age adults in the United States, fewer than 40% are employed. While significant effort towards improving prospects for this population have been at play since the Vocational Rehabilitation Act of 1973, fundamental accessibility issues persist. Considerable legal and technological effort has gone into providing (and mandating) access to interfaces necessary for engagement in education and the workforce, including alternative-format printed materials and magnified or speech-enabled computer interfaces. But one “interface” that neither policy nor technology have addressed is that of daily social interaction.

Individuals who are blind or legally blind remain disadvantaged when it comes to engaging in social interaction, particularly of the serendipitous variety, in which encounters are unplanned and opportunistic (such as after a class, in the lunchroom, etc). While blind/low-vision individuals may passively listen for familiar voices, they are less able, if at all, to actively detect familiar individuals in proximity. This imposes a subtle but real limitation in their ability to form relationships and collaborations as they often find themselves at the mercy of circumstance, waiting for others to approach and identify themselves.

We take this difficulty in actively identifying others with whom a blind/low-vision wearer has previously met as motivation for a system that provides timely and socially-unobtrusive information about so-called proximate acquaintances. Such a system must understand when and how best to provide the wearer with information that does not dis-

tract from ongoing interactions, and must be able to curate training data on new people with whom the wearer interacts without requiring help from sighted persons or burdening the wearer with too many system maintenance tasks.

1.2 Wearable Computing and Social Interaction

Most work at the intersection of wearable computing and social interaction focuses on user activity or social role recognition via gaze/attention tracking through egocentric vision and eye-tracking data. In [10], a wide-angle camera is affixed to bill of a cap and social interactions are classified via HMM as dialogue, discussion, monologue, walking dialogue or walking discussion. [40] uses a head-mounted camera and wearable eye-tracking glasses in combination with random forests to determine when eye contact is established between conversation partners. [27] analyzes videos of ongoing social interactions to determine social roles (e.g. instructor / student, ...) by conducting variational inference on Conditional Random Fields over features including dense HoG, inter-frame person movements, gender classification, RGB-histograms taken on people to capture clothing and interpersonal interactions.

Much less work is concerned with the use of wearable computers that are used to facilitate or augment social interaction. One of the most relevant, though nascent, works in this space is Digikits [7], a concept wearable system separately worn by two people that permits them to discreetly convey small messages to one another while in conversation with yet other people, without the notice of those other participants. In it, the authors qualitatively compare different input (e.g. on-body buttons, dual-purpose gestures) and output (e.g. tactile displays, vibration) methods with regard to how noticeable they might be to others. Our work builds on some of these ideas but is differentiated in that the wearable system provides information about the wearer's environment that they might otherwise be unaware of, and it seeks not so much to hide the communication from others as it attempts to minimize distraction from an ongoing social interaction.

Our own work is distinguished from work in wearable computing and social interactions in that it focuses on multiple data sources (vision, audio, motion) and uses these as

part of a continuously active, end-to-end system that assists (potentially blind) wearers in social interaction.

Chapter 2

Wearable System

2.1 Overview

We set out to design a wearable system that can learn and assist with a (potentially blind) wearer's social interactions. Its intended utility is to unobtrusively inform the wearer about the presence of nearby acquaintances. Three scenarios we specifically design for are:

1. Spotting proximate acquaintances that are passing by or in a crowd, such as when the wearer is walking between classes/offices, and identifying them in time for the wearer to initiate a social interaction.
2. Quickly identifying proximate acquaintances that have just approached or already engaged the wearer so that the wearer can respond with the appropriate level of familiarity.
3. Alerting the wearer about the presence of a specific acquaintance, as when the wearer intends to meet an acquaintance at a particular time and location.

Each scenario requires that the system be able to accurately distinguish acquaintances from non-acquaintances, and also among others or in a group. They further require that the system be able to communicate information to the wearer in a timely and private manner. In the case of scenarios one and two, the information to communicate is in the form of a full name while, in the last scenario, any simple, distinct notification e.g. a tone or buzz,

is sufficient, assuming the correspondence between notification and acquaintance is known to the wearer a priori, though approach direction would also need to be conveyed.

Notification of full names is accomplished by broadcasting an audio message over speaker or headset, or sent wirelessly to a wristwatch, which can communicate the information of an acquaintance either by glanceable text display or vibration-encoded messages. Notification of a single identification is accomplished with a short vibration to the wristwatch.

Identification of acquaintances is handled by face detection and recognition. Face detection is a standard and well-known problem, and relies on models that can be trained offline, prior to system use. Chapter 3 details the training and performance of one such fast and scalable model. Detected faces are then normalized for contrast, size and pose before being fed into facial recognition.

Facial recognition of acquaintances requires that a model be trained in which multiple faces of each acquaintance are present: more are generally better but, empirically, three to six images are enough for practical benefit. Whereas the traditional approach to training a facial recognition engine would involve displaying images of unknown acquaintances to the wearer, in this work, we assume that the wearer may be blind and thus unable to visually identify such images. This constraint, combined with the desire to build a self-contained wearable system that learns the acquaintances of its wearer solely from queries about relevant sensor data, was the motivation for our novel work on the detection of egocentric social interactions, discussed at greater length in Chapter 3.

While going about a typical day, a wearer will interact with acquaintances and non-acquaintances alike. The former, they will want their system to subsequently be able to identify, whereas the latter (e.g. a barista) may not be so important. Our wearable system is designed to detect each of these social interactions and to later query the wearer about each. This is presently done offline, but could be accomplished online at an appropriate time as the feature extraction and learning algorithms are fast to run.

At the end of each day, the wearer is presented with an offline summary of their daily social interactions and asked to label the people with whom they would like their wearable to later recognize. Because the wearer may be blind and thus unable to see the images,

s/he is instead presented with recorded audio snippets from the each social interaction, ideally enough for identification but not so much as to provide significant details about the exchange. This is accomplished by permitting a finite number of audio samples, each of finite length, and is motivated by respect for the privacy of others.

Figure 2-1 visually depicts the workflows and important algorithmic components of our wearable system. The Face and Interaction Models represent the models used for facial and social interaction detection, respectively. In practice, each are trained once, prior to system use, and rarely updated. The Acquaintance Model, in its current form, is simply a face recognition model, and is updated frequently. While acquaintances are recognized in real-time per the green workflow, the current work performs Interaction Detection, and the subsequent user-labeling of audio subsamples (abstracted with the User Labeling step) for the regular updating of the Acquaintance Model, offline. Chapter 3 discusses the training of each model.

2.2 Design Considerations

Our system is built to help its (potentially blind) wearers in social interaction. But we cannot hope to do so if the system is physically or socially uncomfortable to wear. If the wearer finds it burdensome to travel with the system because it's too heavy or because exposed wires encumber their range of motion, then they simply won't use it. Similarly, if others find it awkward or difficult to engage the wearer because the system is too prominent, or they believe themselves to be recorded, or that otherwise natural interactions are interrupted with notifications, beeps or user fumbling, then there will not be people for the wearer to interact with!

At the same time, we need to ensure a reasonable angle of view for an egocentric camera such that it captures good detail on people, both in the area, and with whom the wearer might currently be interacting. We must minimize any unnecessary motion that the camera or accelerometer might be subjected to on account of standard wearer movements like walking and sitting. Audio data must be of sufficient quality that the wearer's voice, at least, can be understood when later reviewed by the wearer. There must be adequate

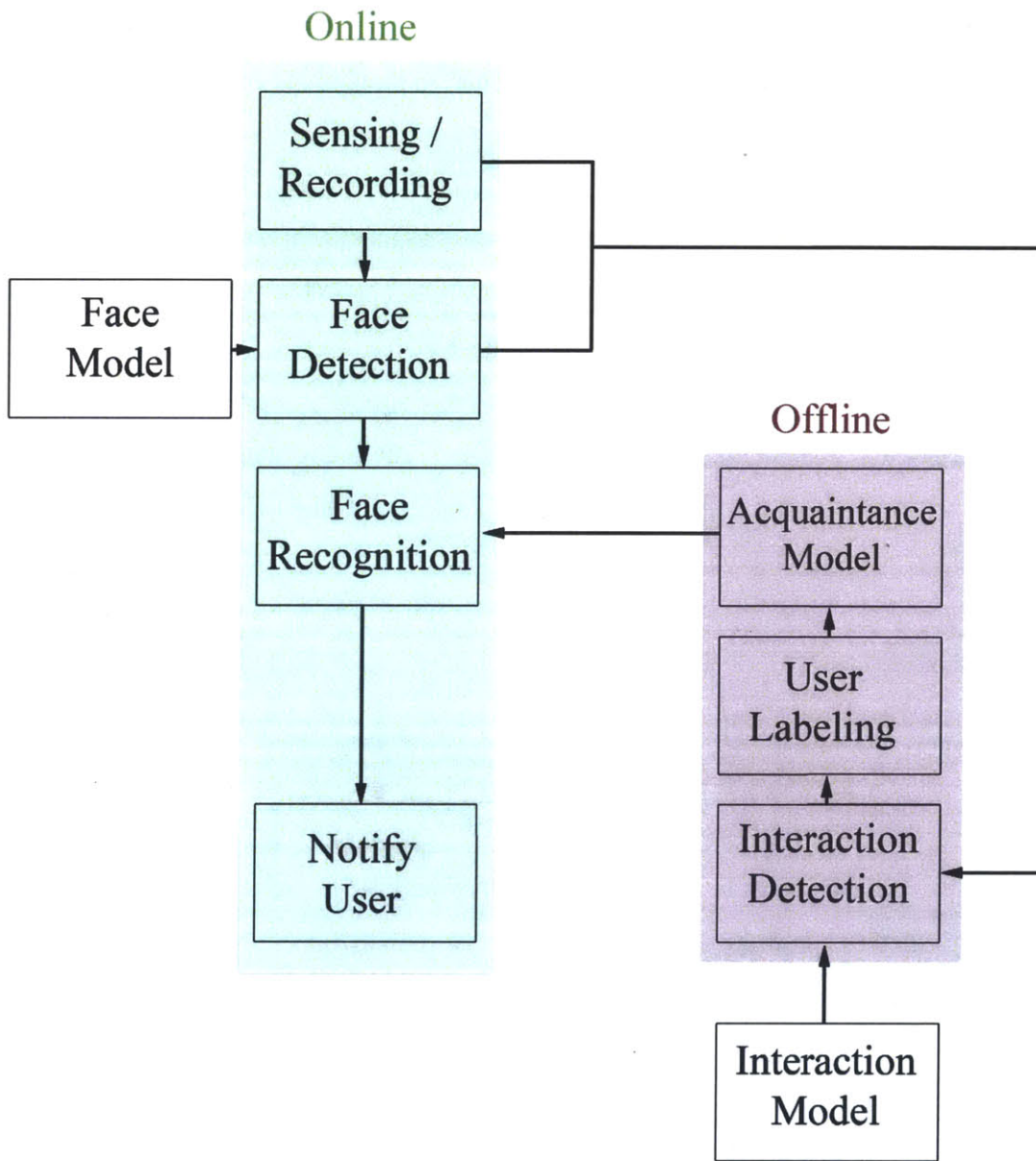


Figure 2-1: Workflows for our wearable system. Boxes in white are models that we build and train prior to system use; they are updated infrequently. The green workflow is performed in real-time whereas the red workflow is performed offline, at the end of each day of use.

computational power to perform rapid detection and identification of nearby persons, and a means to inform them quickly and discreetly.

In determining camera placement, we consulted models that depict how egocentric cameras are affected by viewing angle and wearer motion during a walking movement,

available from [24]. Shown in Figures 2-2i (angle of view) and 2-2ii (motion), we have articulated models of the human form upon which a wearable vision sensor can be continuously placed (at a set distance from the body; in this case, 37.5mm).

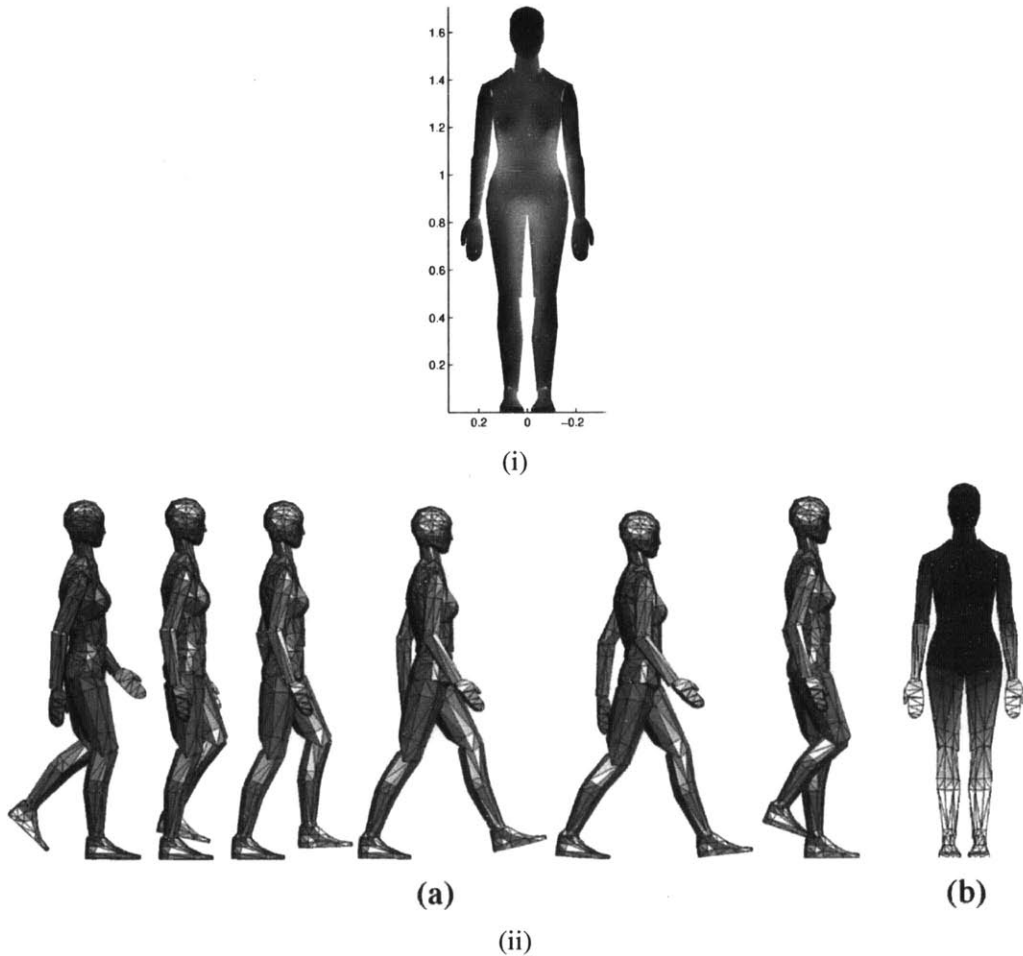


Figure 2-2: Human models of wearable camera angle of view (i) and wearable camera motion (ii). In Figure (i), angle of view is computed by casting rays 37.5mm from each polygonal face of the body in all directions and rejecting those that are obscured by other parts of the body. While all placements will be obscured by the body behind them, some will be more obscured in outward views than others. In Figure (ii), (a) shows motion at each step of a human walk while (b) shows the cumulative motion that a wearable camera is subjected to throughout a complete walk. Darker regions are more desirable and mean greater field of view (i) or less motion (ii). Figures adapted from [24].

In Figure 2-2i, angle of view for a camera is determined by casting rays from each point (with the body in a standing position) and calculating the maximum view area possible by

discarding rays that intersect with other parts of the body. No vision sensor will be able to look backwards, of course, but even so, not all camera placements are equal. In the figure, darker polygons correspond to greater angle of view and thus fewer expected occlusions. We note that the head, shoulders, arms, outer legs and feet have the greatest possible views, followed by the breast, chest and waist, all tailed by the stomach, waist and groin.

In Figure 2-2ii, the motion an egocentric vision sensor would be subjected to is determined by simulation of the articulated body taking a full walking step (a), calculating motion at each polygonal face at each of several points along the step, and then summing the total motion experienced throughout the walking step by each polygonal face (b). In the figure, darker polygons correspond to areas in which a vision sensor will be subjected to less motion. We see that the human face experiences the least motion, followed by the head, chest and stomach, then the upper forearms followed by the upper legs, and finally the extremities.

We additionally considered the social acceptability of wearable camera placement by querying the public. In particular, we asked 100 Mechanical Turk workers to rate the image of one of four subjects (three male, one female) wearing cameras placed on the ear, chest, top of the head and face. We additionally used a control picture for each participant, in which no cameras were worn, and a picture of a large colored pinwheel on the chest, for comparison.

Images of these configurations can be seen in Figure A-3 (appendix), and results in Figure 2-3. We see that it is most acceptable to wear a small camera on the ear, followed with considerably reduced acceptability on the chest. Wearing cameras on the head or face are least acceptable, and invoke more ire than does wearing a large, rainbow-colored pinwheel across the chest!

Considering angle of view, motion subjugation and social acceptability of an egocentric camera, in combination with prototyping constraints, it was determined that although the head would be ideal for angle of view and camera motion, it would be a difficult location for a camera because the wearer and many around them would likely feel uncomfortable during interaction. It would also require that we run cords between the camera and other components, and thus risk physical encumbrance.

Ultimately, it was decided that the camera would be placed upon the chest. Ensuring that it was flat would enable it to lay flush with the body rather than protrude uncomfortably. Obscuring all but the lens would further diminish its presence. Using a smartphone for its camera, audio and accelerometer sensors in addition to its processing, networking and built-in battery, would mean that no backpack need be worn, or wires run. The primary drawback, then, would be the limited ability to control for camera angle of view, as most smartphones have relatively standard, non-wide angles of view, and options for a slender, high-quality external lens were not available. This drawback was deemed acceptable in the context of a prototype system.

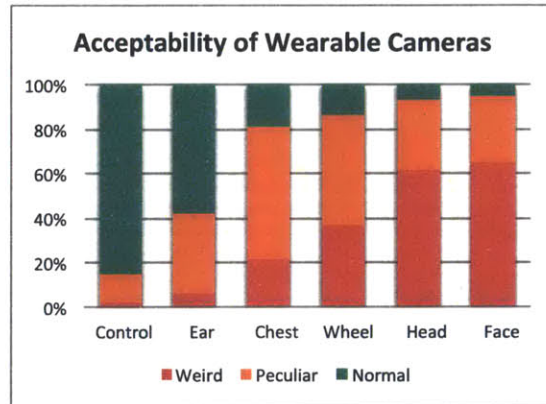


Figure 2-3: How does the public feel about wearable cameras? We asked 100 people to rate the social acceptability of different wearable configurations worn by four people. Our results (bottom) suggest that the best wearable camera may be the one that is not seen.

2.3 Physical Design

Our wearable system involves a custom jacket or lanyard, inside of which is a modern Android phone equipped with the standard array of triaxial accelerometer, microphone, camera and wireless (cellular, Bluetooth and WiFi) connectivity. The design of the jacket or lanyard is such that the smartphone is covered save for the camera lens, which peers out chest-level from the wearer. Connected via Bluetooth is a wristwatch capable of programmable text display and text-encoded vibration for user output. Also contained on the

watch are three hardware buttons, for user input. Each watch button is capable of detecting single, long and repeated presses, with programmable delays for the latter two options. These configurations can be seen in Figures 2-4i and 2-4ii.

Because the watch is worn on one hand and manipulated with the other, it is sometimes preferable to additionally have a slender Bluetooth remote contained within the wearer's pocket, with buttons facing outward such that the user can quickly and discreetly press them without placing their hand in the pocket.

An additional optional accessory is a Bluetooth headset so that the wearer may receive private audio messages rather than having messages quietly broadcasted by the speaker, sent to the watch display, or encoded as vibrations.



(i) Jacket Configuration

(ii) Lanyard Configuration

Figure 2-4: Our wearable system in two configurations. It is most discreet in the Jacket Configuration (a), but most easily donned and worn by different people in the Lanyard Configuration (b).

2.4 Hardware

Our wearable system incorporates the following components:

1. Jacket or Lanyard

The Jacket or Lanyard are interchangeable and used for securing the smartphone to the wearer's person such that its camera points outward at chest height. The phone and camera have a portrait orientation for physical comfort and so that human faces tend to be in full view when the wearer interacts with them at normal interaction distances (3ft - 10ft).

Whereas the Lanyard is simply a hole-punched leather pouch suspended around the neck, the Jacket is more involved because it requires stitching and surging a form-fitted pocket on the interior lining of a dark jacket, with a small hole through which the camera can discreetly peer. Schematics for this are shown in Figures A-1 and A-2 in Appendix A.

2. Smartphone

The smartphone is used as a prototype wearable computer; that is, users do not interact with it as though it were a phone. In fact, its screen is never powered during normal use.

The phone is used for its image, audio and accelerometer sensors, for local computation (including face detection), and network connections (WiFi or cellular for connecting to a server to offload processing) and Bluetooth for sending/receiving data from the watch, remote, or headset.

We use the Android platform, chosen for its native ability to indefinitely run programs without the screen being powered on. Our particular model is the Samsung Galaxy S2, chosen on account of its smaller footprint (9.7mm thickness, 116g mass), for its reasonable camera image quality at 800 x 600 preview resolution, and for the ability to replace its battery as needed so that the system can be used all day. Used in portrait orientation as we do for our work, this model has a somewhat limited angle of view of 46.3° horizontal and 59.6° vertical, with a focal-length of 3.97mm.

3. Smartwatch

Used for handling user-input via button presses and providing user-output via glanceable text display or text-encoded vibration. The watch thus acts as a sort of dumb wearable terminal, but is programmed so that the wearer can control the state of the wearable system with button presses.

We use the Pebble Smartwatch, pictured in Figure 2-5 for its programmable, embedded-C environment with 144 x 168 resolution e-ink display, programmable vibrating motor, three programmable physical button inputs, Bluetooth radio, which remains in connection with the smartphone, and week-long battery life.



Figure 2-5: Watch used in our wearable system for user input (with the three buttons on the right) and output (via glanceable text display or vibration-encoded text).

4. Pocket Remote (optional)

Optionally used for discreet, one-handed input, but restricted to signalling. It remains in a jacket or pants pocket with buttons facing outward so that the wearer can press them through the outside of their clothing.

We use a Satechi BT MediaRemote, pictured in Figure 2-6 (right), configured so that all buttons perform the same function (so that the wearer doesn't need to attempt precise inputs from within or outside the containing pocket).

5. Bluetooth Headset (optional)

Optionally used for personal audio output to the wearer. This is the fastest communication medium available on our wearable system but is not typically used for the perceived negative effects its presence has on the wearer's social interactions. We used the Plantronics M155, shown in 2-6 (left).



Figure 2-6: Optional accessories for our wearable system. Left: a Bluetooth headset for personal audio. Right: a Bluetooth remote placed within a pocket and used for discreet cueing. All buttons are mapped to a common function so that the wearer need only press against the outside of their pocket.

2.5 System Architecture

Our end-to-end system consists of the wearable system and a server. At the heart of the wearable is the smartphone, which runs all local sensing, processing, network activity and logging. Connected to the smartphone via Bluetooth is the smartwatch and either of the optional accessories (headset, remote). The smartphone handles user inputs from the watch or remote, and drives user output via the watch or, optionally, the headset or its own speaker. The server runs a set of scripts so that it can receive online requests from the wearable (such as for acquaintance recognition) through an Internet connection, and is also used to keep the Acquaintance Model current by running and combining social interaction detections on each day's wearable data with user-supplied labels of the people they interacted with (obtained through audio clips).

The smartphone runs a custom application written for a standard Android 4 kernel. Although Android makes background processes available as so-called Services, restrictions placed upon them (such as sensor and peripheral access) make it more convenient to write the application as a foreground process, called an Activity in Android, but with the screen powered off. This reduces battery life, but improves ease of implementation, particularly as ours is not the intended use of the standard Android platform.

The application can conceptually be thought to consist of three components: Handlers, Modes and Preferences. Modes are states that the wearable can be configured to be in by the user at runtime. Handlers are services that are either actively running regardless of the current Mode, or else services that any Mode has the ability to control. Preferences are a simple way of storing persistent flags and options like server address or bandwidth management. Figure 2-7 shows the (simplified) layout of each of these components.

Handlers are built to handle the basic functionality that's shared by different Modes. They are built so that Modes can be easily written or modified. User-input Handlers exist for each peripheral but, more typically, a programmer subscribes to the coarse-grained RemoteControlHandler, which itself handles input from each peripheral and abstracts them into a common interface. Handlers are available for each sensor (camera, audio, accelerometer, battery-status) so that a Mode need only activate them synchronously and then subscribe via callback to their (asynchronous) outputs. Higher-level functionality is also built into Handlers, such as the NetworkHandler, which builds and maintains the system's wireless connections if they're broken. Notably, the UserOutputHandler is used to abstract the way that information is conveyed to the user—a Mode need only provide the text of the notification and it will be delivered to the user by glanceable text, vibration encoding, or audio, as necessary. Although the system presently makes these output options available to the user as a Preference, in the future, this will be handled by online detection of whether the user is engaged in a social interaction.

Modes are high-level states that the wearer can put the system into. By default, the wearable is in the NoneMode, in which no external network connections are opened, nor any CPU-bound processes run. The system is effectively at rest, with a few Handlers periodically running processes that ensure that the watch remains connected, and keep system

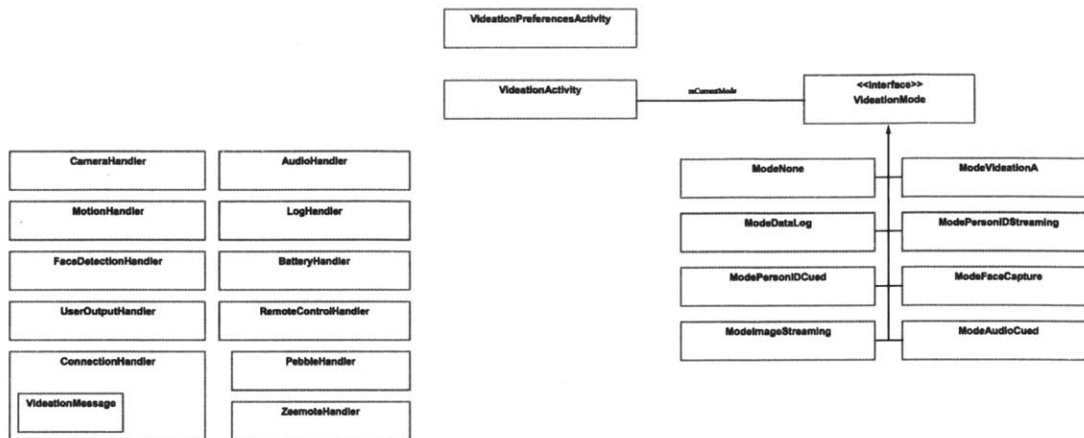


Figure 2-7: Software architecture of our wearable system. The various Handlers (left) are available to each user-visible Mode, which itself inherits from a VideationMode interface. This design makes it easy to implement new or change existing Modes.

status (e.g. battery life for both the Smartphone and Smartwatch components) available to the user.

The typical Mode for active use is the VideationMode. In it, camera images, microphone audio, and accelerometer data are actively logged, and a connection with the server is maintained through the Internet via WiFi or cellular network. Images are analyzed in real-time for faces (either locally or remotely), and faces are checked against the Acquaintance model (remotely), with user notifications made by audio, glanceable text and/or vibration encoding (at the user’s option). Offline, scripts are available on the server that sync system logs and sensor data, perform social interaction detection, and instantiate a blind-accessible interface in which users can label the identities of people they interacted with by listening to audio clips from each interaction.

Other Modes can be written as well, with a minimum of effort thanks to the Handlers and Preferences. In one, user-input cues the system to attempt synchronous detection and identification of a nearby acquaintance (rather than doing so continuously). In another, user-input cues audio clips to be recorded and sent to the server, which stores and attempts to transcribe any proper nouns (such as a persons name, given during an introduction).

2.5.1 Network Connections and Data Transport

Lightweight Communications and Marshalling (LCM) [13] is one standard robotics package for streaming message passing between systems. It is noteworthy for its low overhead, fast UDP-based connection, static typing of messages and multi-platform capability. Unfortunately, LCM messages cannot easily traverse subnets. In fact, doing so requires that it give up its fast UDP transport so that data can be tunneled via TCP. But even this is an extension to its core code, and not available for all platforms, Android among them.

Porting this ability was considered but so were alternative message passing protocols. Ultimately, the relatively new WebSockets protocol [12] was chosen for handling connections between the wearable system and the remote server. WebSockets is a TCP-based protocol built for bidirectional communication of data sent over port 80 connections established via standard HTTP handshakes. It is distinguished from previous HTTP-based methods in that it formalizes the idea of persistent Internet-based connections rather than using workaround techniques such as "long-polling," in which an HTTP connection is made to stay alive indefinitely by configuring the client and server to continuously send small messages.

Because WebSockets uses standard HTTP handshakes, it can easily traverse typical network topologies including through proxies. The ability to communicate between subnets is essential for our wearable system as the wearable may not always be on the same network as the server, such as when the wearer leaves the building and the system connects to another WiFi network or falls back to cellular access.

But, unlike LCM, WebSockets does not itself handle message serialization. For this, we considered several alternatives: chiefly, the serialization component of LCM, Google's open source Protocol Buffers¹, and MsgPack². All serialize binary data and permit pre-defined, statically-typed messages. We devised a simple experiment in which our wearable system streamed VGA-resolution, JPEG-encoded images (with timestamps) to the server, and marked average encoding and decoding times, as well as the size of the packed message. Table 2.1 shows the results and Figure 2-8 shows the definitions of each message.

¹<https://developers.google.com/protocol-buffers/>

²<http://msgpack.org/>

Averages were taken over multiple seconds of data streaming.

```
// lcm definition
package lcm_jacket;

struct jpg_t {
    int64_t    utime;
    int32_t    nBytes;
    byte       jpg[nBytes];
}

// protobuf definition
message jpg_t {
    required int64 utime = 1;
    required bytes data = 2;
}

// msgpack definition
msgpack_jpg_t {
    public byte[] data;
    public long utime;
}
```

Figure 2-8: Definitions for testing object serialization in the LCM, Protocol Buffer and MsgPack Object Serialization libraries.

We see that LCM sends messages of more than double the size relative to either Protocol Buffers or MsgPack. It is considerably faster at encoding than either alternative, and nearly fastest at decoding. Assuming, however, that we plan to transmit 30 frames per second, we note that the total time spent encoding and decoding LCM messages is 2.19ms, compared to 8.37ms for MsgPack messages, and 4.26ms for protocol Buffer messages. Similarly, total network bandwidth for LCM messages is 1.84MB, compared to 0.77MB for MsgPack messages and 0.84MB for Protocol Buffer messages. Considering that our

	lcm	protobuf	msgpack
Packing Size (bytes)	64250.98	29235.71	27075.43
Encoding Time (ms)	0.057	0.111	0.267
Decoding Time (ms)	0.016	0.031	0.012

Table 2.1: Data serialization sizes and encode/decode times for streaming VGA-resolution JPEG images along with a timestamp in three modern serialization libraries. Data is encoded on a dual-core Android platform and sent over the WebSockets protocol to a quad-core Linux server.

wearable system will sometimes be limited to upload speeds of less than 1Mbps on slow 3G cellular connections, it makes sense to minimize message sizes. But, protocol Buffers offer additional conveniences beyond MsgPack that are worth the small increase in typical message size (and partially made up for by slight improvements in encoding/decoding time).

```
message VideationMessage_t {
  message ImageJpg {
    required bytes data = 1;
  }
  message Audio3gp {
    required bytes data = 1;
  }
  message SystemStatus {
    required float battery_percent = 1;
    required string wifi_ssid = 2;

    required bool pref_streaming_doHandshakeBetweenFrames = 4;
    required bool pref_camera_greyscale = 5;
    required bool pref_mode_person_id_stream_type_is_faces = 6;
  }
  message PersonIDResponse {
    required int64 utime_query = 1;
    optional string results = 2;
  }

  required int64 utime = 1;

  optional string mode = 2;
  optional SystemStatus system_status = 3;
  optional ImageJpg image_jpg = 4;
  optional Audio3gp audio_3gp = 5;
  optional PersonIDResponse person_id_response = 6;
}
```

Figure 2-9: Serialization definition for the canonical VideationMessage, which gets passed between the wearable system and server, with fields filled in as appropriate. Note that the only required field is the timestamp, and that only the fields necessary for a given task are transmitted (per the optional specifier).

Particularly useful within the Protocol Buffer standard is the ability to use nested class definitions in combination with optional data field specifiers. Taken together, this allows a single, canonical message to be defined, with only those optional data fields that are relevant to the message filled in at any given time. We use this ability to good effect by making a

single `VideationMessage`, shown in in Figure 2-9, that defines the format for messages passed between the wearable system and the server. The smallest message contain only a timestamp, as when a heartbeat is being sent. But in other cases, the wearable system will attempt to transmit a `SystemStatus` message whenever the wearer changes the system's mode. In its current form, the server responds only with `PersonIDResponse` messages, which are returned only when it receives an `ImageJpg` message and previously knows that the system is in a `Mode` that involves person identification (established by a previously-sent message).

2.6 User I/O

User input is handled by the wearer pressing any of the three buttons (top, middle, bottom) on the watch. A long-press on the middle button cues a `Mode-select` menu whose options are acquired from the smartphone at runtime. This menu is scrolled through via the top and bottom watch buttons, and selected with the middle button. Each `Mode` has access to the three buttons and may use them however desired (with the exception of the middle long-press). It is worth noting that although the default method for navigating this `Mode-select` menu is by viewing the text on the watch screen, it is easy to augment it so that options are made available to a blind wearer via audio or vibration encoding. In the standard `Videation Mode`, the middle button is used to 'arm' or 'disarm' sensing for nearby acquaintances and logging sensor data for offline social interaction detection. The top button is used for marking events of interest in the system logs, and the bottom button is used to repeat the last notification. The optional remote peripheral is treated as having a single button input equivalent to a single-press of the watch's middle button.

Output is typically made available to the wearer on the watch. This can either be through text on its display, which can be quickly glanced while traveling or interacting with others, or by encoding the text through vibrations to the wrist. As an alternative, audio can be delivered by speaker from the smartphone for the wearer and any nearby person to hear, or through the headset.

While glanceable text and audio output are straightforward ways to convey information

to the wearer, they are not without drawbacks. Glanceable text requires that the wearer raise their hand and cast their eyes down. Audio requires that the wearer broadcast their use of the wearable either through wearing a Bluetooth headset (still considered socially undesirable in many circumstances) or else use speaker audio, in which the existence of their system is made plain to others. As this work intends to assist the wearer in social interactions, and these output methods can negatively impact them, we have opted to explore alternatives, in particular, Morse-Coded messages delivered by vibration to the wearer's wrist (via the smartwatch).

2.6.1 Vibration-Encoded Messages

The International Morse Code Standard [2] defines the dit as its atomic unit of time. There is a short and long pulse (measured as one and three dits, respectively), as well as silent gaps between pulses, characters, and words (measured as one, three and seven dits, respectively). Words per minute (wpm) can be measured by the number of times that the word 'PARIS' can be encoded in one minute (including the gaps between pulses, characters and words). The specific word 'PARIS' is often used because it is representative of plain English text with a normal distribution of characters.

Commercial radiotelegraph licensure in the United States requires that applicants be capable of reading Morse code at 20wpm or 25wpm, depending on the license class, with anecdotal evidence suggesting that expert rates hover around 35wpm, and record rates around 75wpm for veteran operators. To put these rates in perspective, Figure 2-10 shows the time to encode an average English letter (weighted by English usage or not) plotted against encoding speed. We see that even at expert levels, each weighted character takes about 150ms to encode. But, remembering that there is a single-dit of silence between each pulse, and that the dit is, for instance, 60ms in duration at a rate of 20wpm, we see that encoding a simple word (such as a name) takes nearly 2 seconds, even at expert speeds. Figure 2-11 shows these encoding times for different encoding speeds.

Informal experimentation suggest that 15 – 20wpm vibration encoding speeds are discernable and appear to be a reasonable training goal, but that such training takes regular

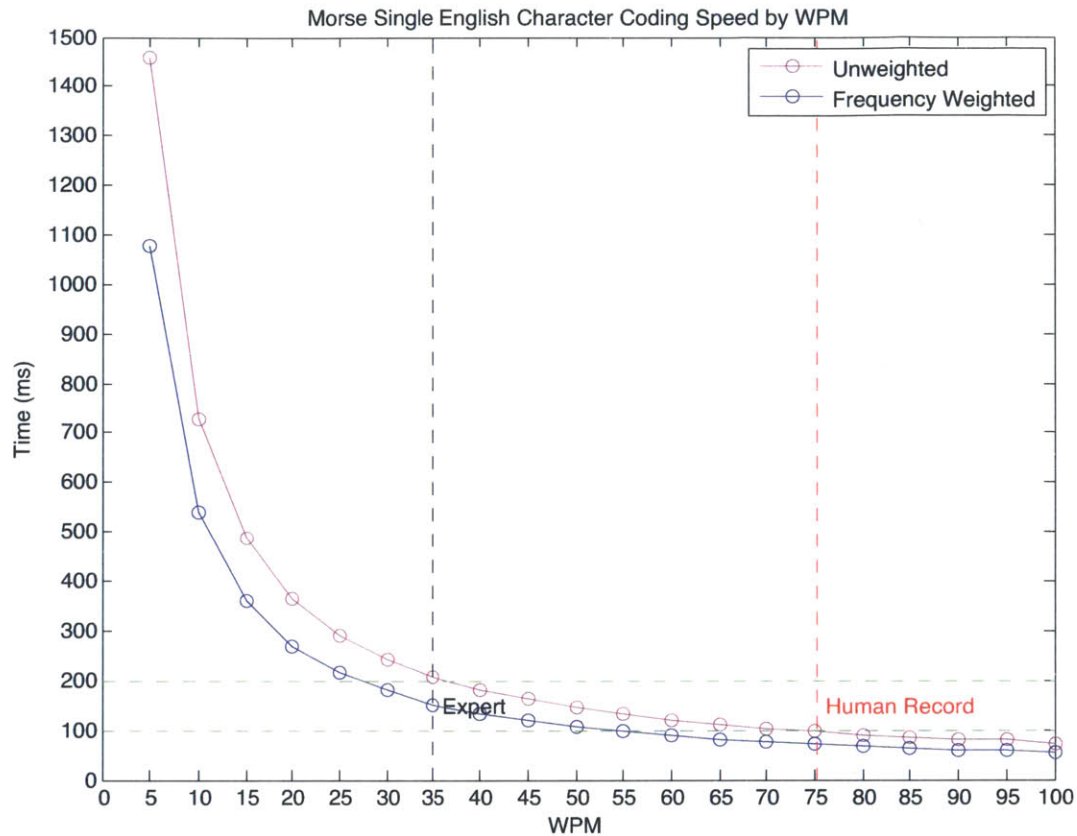


Figure 2-10: Frequency-weighted character encoding time with expert and record human Morse-decoding performances.

practice for a period of several days to weeks. It appears to be facilitated by the use of spaced repetition learning [37] techniques, in which encoded letters (and later n-grams) are cued for the wearer to discern based on past wearer-graded performance on those same letters.

For perspective on these vibration encoding speeds, Audiobooks are recommended to be recorded at 160wpm [36] for simple content and English speaking rates have been found to range between 120 to over 220wpm [31]. Experienced, blind users of synthesized speech have been found to be capable of comprehending synthesized speech at as much as double the rate to those of less-experienced, normally-sighted users [25].

Thus, while a wearer might reasonably train to 15-20wpm vibration encoding speeds, they are still stuck waiting three seconds for the complete notification of a proximate ac-

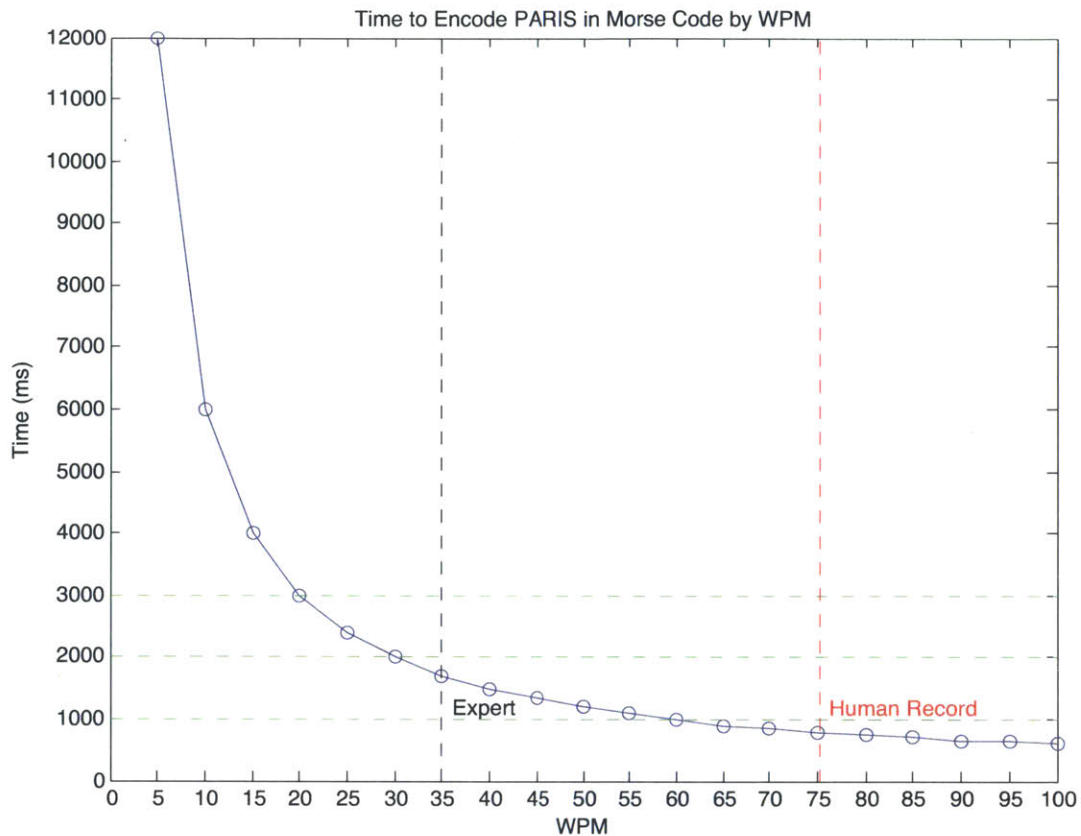


Figure 2-11: Word encoding time (for PARIS) with expert and record human Morse-decoding performances.

quaintance, as opposed to synthesized speech encoding times that, without training, will be easily understood at a 375ms encoding speed. With training (that they may already have) this could conceivably be brought to as little as 150ms.

The advantage that vibration encoding has over audio notifications, then, is that it can be delivered privately and ignored with minimal disruption to the wearer if desired. A middle ground that would bridge the differences between encoding speeds could be to establish correspondences of peoples names to small letter groupings, so that Chris might be encoded as c or ch, which would bring encoding speeds down to well under a second (see Figure 2-10), even in the case of 15-20wpm, and would be competitive with audio notifications at expert encoding speeds of 35wpm.

Bone conduction (discussed and studied in e.g. [35]) is a promising option that could provide the fast, audio-based notifications of headsets along with the relative privacy of

vibration-encoded messages. Here, sound vibrations cast directly upon bones near, but not in, the ear can be heard by a person, without obstructing their own aural perception of their environment. If the audio could be conducted well enough that it wasn't heard by others at a conversational distance, and the headset could be made small enough and appropriately placed (perhaps behind the ear), then the wearer could have fast, private notifications that would require no learning curve on their part. Unfortunately, while recent products on the market have claimed to be using bone conduction, many seem to functionally be more aptly described as a small speaker located near the ear, in plain auditory and visual view. Perhaps this will change in time.

2.7 Detection and Recognition

Our wearable system uses different algorithms for person detection depending on whether it's done online, for acquaintance recognition, or offline, for social interaction detection.

In the online case, the system can either be configured to do detection locally, on the wearable platform, or remotely, by the server. If done locally then Viola-Jones [34] is used for its efficient implementation (15fps on our platform at VGA resolution) and acceptable, if modest, performance. If done remotely, then either the AUC-boosted cascade detector discussed in Chapter 3 can be used, or an even more reliable cloud-based detector [1] can be used. Benefits of detecting locally include the ability to save bandwidth by only transmitting detections, rather than streaming images, and would ideally reduce the time it takes to recognize and communicate the identity of an acquaintance to the wearer. But on this latter point, we unfortunately find that the local detector requires such precise frontal poses that it's often faster to stream images to the server and detect it there.

For offline detection, the system can use any of the online options, or the slow (10s) but high-quality detector from [41], which also provides pose estimation and landmark (eyes, nose, mouth) estimation.

Recognition is handled by the same cloud service that is capable of performing online or offline detections. Empirically, its results are better than that of standard, available methods such as Eigenfaces or Fisherfaces [4]. Part of this is due to the need to feed face

recognition engines with not just detected faces, but to also have those faces accurately aligned to a frontal view with a pose estimator, which is difficult to do online.

2.8 Logging

Our wearable system maintains text logs of Mode changes and diagnostic information. When recording sensor data, it stores images as separate images in a directory, audio as *.3gp* encoded files, and accelerometer data (corrected for gravity) in CSV files. Image and audio files are named with the timestamp of the time the sensor data began to be collected (not when the file was made), making it feasible to manually synchronize them post-hoc, as is necessary for social interaction detection.

In an ideal implementation, social interaction detection would be conducted online so that the sensor data could be discarded upon analysis for interactions or acquaintances, thereby protecting the privacy of the wearer and nearby persons.

2.9 User Labeling of New Acquaintances

Offline, such as at the end of each day, the wearable system syncs its logs and sensor data to the server and social interaction detection (discussed in Chapter 3) is performed. The wearer is provided with audio snippets from each interaction and asked to identify the person(s) with whom they interacted, or to label them as not presently needing to be identified in the future (though it would raise the question again if the wearer interacted with them on another day).

Chapter 3

Detecting Social Interactions

3.1 AUC-Boosted Person Detection

3.1.1 Introduction

We seek to build a high-performance, video frame-rate face detector for embedded platforms. Unfortunately, while the well-known Viola-Jones detector [34] is widely available, it is non-optimal because:

1. Viola-Jones cannot run at video frame rates on high-resolution images, which limits the effective size, distance and latency with which objects can be detected. This is partly due to the many boosting stages (24 in OpenCV's reference implementation) that it employs.
2. Adaboost can take weeks to train when using several hundred thousand Haar-like features and simultaneously meeting specified maximum false-positive and minimum true-positive thresholds, even when done on server hardware, making it painful to continuously integrate new data.

Since Viola-Jones' popularization ten years ago there have been substantial improvements in discriminative, fast-to-collect features that enjoy some combination of scale, rotation and lighting invariance, among them SIFT [22], SURF [3] ORB [28] and Histograms of Oriented Gradients (HOG) [8]. In this work, we seek to exploit their improved discriminative

ability to build a smaller (in number of boosting rounds), faster face detector that’s quick to train and which outperforms stock Viola-Jones.

In particular, we implement from scratch an adaptation of the approach advocated in [21], which itself is inspired by the Viola-Jones framework, but significantly modifies it in the following ways:

1. Rather than jointly satisfying a maximum false-positive and minimum true-positive rate for convergence at each boosting stage, the AUC criterion is used which, for the t^{th} boosting round is defined as:

$$H^t(X) = \arg \max_{k=1:K} J(H^{t-1}(X) + a_k h_k(X), X)$$

where $H^t(x) = \sum_{t=1}^T a_t h_t(x)$ is a weighted combination of weak classifiers and $J(H, X)$ is the area under the ROC curve produced by classifier H on data X .

2. Rather than use decision trees for the weak classifiers, binary logistic regression is used so that $P(y = \pm 1|x, w) = \frac{1}{1 + \exp(-yw^T x)}$ in which parameter w is found by minimizing the regularized log of the above.
3. SURF features are used as opposed to Haar-like features, which dramatically reduces the number of tested feature combinations required from several hundred thousand to several hundred per template.

With these changes, the authors claim improved performance 0.3 greater true-positive rate (TPR) for a given false-positive rate (FPR) on a modern dataset [17] over Viola-Jones while enjoying an approximately $2x$ runtime speedup and a massive training time speed-up from multiple weeks to under an hour for over one billion training samples.

In Section 3.1.2, we discuss our adapted use of extended SURF features. In Section 3.1.3, we discuss the dataset we collect. In Sections 3.1.4 and 3.1.5, we discuss our use of logistic regression as a weak learner, and detail the AUC boosting algorithm that brings everything together. In Section 3.1.6, we show encouraging results and in Section 3.1.7 we discuss limitations and future work.

3.1.2 Features

We implement the dense, upright, extended SURF descriptor [3] from scratch. This differs from the standard SURF setup in that there is no interest point detection (features are densely sampled) and the detection template is kept axis-aligned (descriptors lose rotational invariance, chosen for efficiency). The implementation is as follows:

1. Compute image gradients d_x, d_y by convolving the image with kernels $[-1, 0, 1]$ and $[-1, 0, 1]^T$, respectively.
2. Compute and stack eight integral images: $\sum d_x, \sum |d_x|$ for $d_y \leq 0, d_y > 0$ and $\sum d_y, \sum |d_y|$ for $d_x \leq 0, d_x > 0$.
3. For a given template (max size $(w, h) = (40, 40)$ in our case), split into four cells and within each collect, then append, the 8 above sums for a total of 32 array accesses and a 32-dimensional descriptor.

Boosting requires that we have many candidate weak learners, which themselves require different features. We accomplish this by varying the scale (12 to 40), aspect ratio ($(w, h) \in \{1 : 1, 1 : 2, 2 : 1\}$) and offset (4 pixel steps) of the template. For each $(w, h) = (64, 64)$ pixel image patch, then, we have 361 possible sets of 32-dimensional features. It should be noted that there are many more possible features but, unlike Viola-Jones, we don't expect to need to check every possibility due to the increased discriminative capacity of the features.

3.1.3 Data

For positive samples, we use the Labeled Faces in the Wild dataset [14] preprocessed by [29] so that faces are size-normalized to $(w, h) = (64, 64)$ pixels and tightly cropped (but not aligned). In total, there are 13,233 positive samples. This dataset is typically used for recognition rather than detection benchmarking and, indeed, was initially built with a well-tuned Viola-Jones detector. If we split this data into training and testing then we automatically have a goal (with respect to true-positive rate) and means of comparing any classifier we test to an 'optimal' VJ detector.

For negative samples, we use Caltech101 [11] with two of its (human face) categories removed for a total of 99 categories of negative samples. In a preprocessing step, these were chopped into $(w, h) = (64, 64)$ -sized chunks, such that each chunk had no overlap with any other. This produced a total negative sample size of 62,346.

It is worth noting that negative samples wouldn't typically be separated into disjoint chunks, as we did, but would instead be densely scanned. In that way, it is relatively easy to accumulate billions of negative samples. We forewent this strategy for ease of implementation (specifically, to extract features offline and keep them in memory) and to keep computational cost reasonable but as we'll later see, we would need to adopt the more common approach in future work.

3.1.4 Weak Classification

Being (relatively) high-dimensional, our choices for features are not well-suited to decision stumps as are commonly used with boosting. Instead, we use binary logistic regression, in which

$$P(y = \pm 1|x, w) = \frac{1}{1 + \exp(-yw^T x)}$$

where x is a feature patch (32-dimensional in the case of the SURF descriptor) and w is learned by minimizing

$$\sum_{n=1}^N \log(1 + \exp(-yw^T x))$$

We implicitly regularize by implementing the 'dynamic working set' as detailed in [38] (to be described below).

3.1.5 AUC Boosting and Cascade

One of the nice features of the Viola-Jones framework is that it can be tuned to specific performance requirements. To achieve a given true-positive rate (TPR) D with less than F false-positive rate (FPR) in c cascades, then because $F = \prod_{i=1}^c f_i$ and $D = \prod_{i=1}^c d_i$ for f_i, d_i the FPR and TPR of the i^{th} cascade, respectively, we can stipulate a priori that it is necessary for each cascade to achieve $d_i = \exp(\frac{\log(D)}{c})$ and $f_i = \exp(\frac{\log(F)}{c})$ for $i = 1 \dots c$.

For example, to achieve $D = 0.9$ TPR with $F < 1e - 6$ (this latter being a consensus ceiling on a practical detector), one could train a $c = 10$ stage cascade in which each stage must achieve $d_i = .99$ TPR with no more than $f_i = .25$ FPR. But, achieving that may be computationally expensive (because more weak learners will be necessary), particularly in the earlier stages of the cascade, and so one might opt to increase the number of cascades to, for example, $c = 20$, hoping that the overall performance improves. In this case, each cascade would need to achieve a slightly higher true-positive ($d_i = .995$) but is permitted significantly more false-positives ($f_i = 0.5$).

Thus, by enforcing specific TPR and FPR at each level of the cascade, one can model the runtime of the detector. The drawback is that training with these fixed-rate assumptions can create cascades that are unnecessarily complex. One could in principal account for this by training under many sets of candidate parameters, but each test will take several weeks, partly due to the large number of features and possible stumps, and partly because the Viola-Jones formulation requires the joint optimization of TPR and FPR in each cascade.

Given training set $(X, Y) = \{x_n, y_n\}_{n=1}^N$, $y_i \in \{-1, 1\}$, k possible sets of features for each, we seek to learn $H(X)$ with maximum false-positive rate F and given minimum true-positive rate, d , per cascade. The complete algorithm is specified in Figure 3-1.

The basic idea is that we build a cascade of classifiers where, within each cascade, we train candidate weak classifiers on a small, dynamic and balanced subset of the training data called the *dynamic working set*. We select the learner that maximizes the area under the ROC curve (AUC) as evaluated with previously selected weak learners **on the entire training set**. We then weight the newest learner as in Adaboost but with error $1 - AUC$ and after performing standard Adaboost data reweighting, we update our dynamic working set by replacing those positive and negative samples which have the smallest 10% weights with random samples from the entire training set. Learning a single cascade is finished when the AUC has converged (is no longer measurably improving) or when an artificial boosting limit has been hit (so as to limit complexity). Between training cascades, we refresh our dynamic working by evaluating the whole classifier on all training data and choosing those it fails to classify correctly).

Training is complete once we have fallen under a prespecified false-positive rate (for

1. $F_i = D_i = i = 0$
2. Initialize positive and negative data weights: $w_{1,i}^+ = \frac{1}{N^+}$, $w_{1,i}^- = \frac{1}{N^-}$ for N^+ , N^- the number of positive and negative samples.
3. Randomly select a set $Z \subset X$ of balanced positive and negative training samples.
4. while $F_i < F$:
 - (a) $i = i + 1$
 - (b) Learn boosted model $H_i(x)$
for $t = 1 : T$:
 - i. For each of the K possible feature sets, train a logistic regression model $h_k(x, w)$ on Z
 - ii. Add to $H_i^{t-1}(x)$ the weak classifier:

$$\arg \max_{h_k} J(H_i^{t-1} + h_k, X)$$

for $J_t = J(h, x)$, the area under the ROC curve produced by classifier h on dataset x .
 - iii. Set error $e_t = 1 - J_t$, $\alpha_t = .5 \log(\frac{1-e_t}{e_t})$, and update then normalize data weights such that $w_{t,i} = w_{t,i-1} \exp(-y_i \alpha_t H_i^t(x_i))$.
 - iv. Replace with random samples from X those $z_i \in Z$ for which $w_{t,i}$ is lower than 90% of the other $w_{t,j}$.
 - v. break if J_t has converged or prespecified maximum boosting rounds achieved.
 - (c) Evaluate $H_i(x)$ on X , choosing from its ROC curve threshold Θ_i at point (d_i, f_i) s.t. $d_i = d$.
 - (d) $F_{i+1} = F_i * f_i$, $D_{i+1} = D_i * d_i$
 - (e) If $F_i > F$ then evaluate $H^t(x)$ on X and replace correctly classified $z_i \in Z$ with misclassified samples.

Figure 3-1: The AUC-Boosted Cascade Algorithm. Note that we use $H_i(x)$ to denote the classifier resulting from training cascades $1 \dots i$, while $H_i^t(x)$ represents a classifier in the t^{th} round of boosting construction within a cascade i .

practical applications, this starts at $1e - 6$), at which point our true-positive rate is determined from the ROC curve of the boosted cascade.

A key difference between this formulation and Viola-Jones as specified above is that, in our algorithm, training of an individual stage of the cascade is complete when that set of boosted weak learners is no longer meaningfully improving upon the AUC, at which point

a threshold is determined by selecting the point on the ROC curve that corresponds to a pre-specified minimum true-positive rate. Thus, the false-positive rate will likely be different for different stages. Relative to Viola-Jones, the gamble here is that, for a particular minimum true-positive rate, we often (but not always) do better than a particular false-positive rate. If this gamble is successful, our classifier will result in fewer, less complex cascades. Otherwise, it will be longer. We expect the former behavior, however, because we are using more sophisticated features.

3.1.6 Results

We randomly cut our 13, 233 positive and 62, 346 negative samples into 90%/10% training / test split and train two models: one using AUCBoost with SURF features and logistic regression weak learners as described above, and another using untuned, unboosted HOG features (via code from [23]) and a simple LDA classifier. The ROC curve for the test sets of each are shown in Figures 3-2 and 3-4, respectively.

The ROC curve for our AUCBoost method ultimately achieves an AUC of 0.998, corresponding to 0.7 TPR for $1e - 3$ FPR. To see how training error (defined as $1 - J_t$ for J_t the AUC at round t , defined in step iii. of Figure 3-1) varies with boosting round, we look at Figure 3-3. Strikingly, we see that the error falls below 0.01 within 3 rounds of boosting suggesting that our weak learners aren't actually all that weak.

An AUC of 0.998 seems impressive, but when we recall that a practical detector requires no more than $1e - 6$ FPR, corresponding to one false positive per ten VGA-resolution images (assuming 64x64 window size, 5 pixel step, single scale, then we see that $1e - 3$ FPR would yield around ten false-positives per image and would thus not be that effective. Inspection into the training data yielded that this classifier reached 0 false-positives in training and so, clearly, further improvements to FPR would require more negative training samples (which, after the fact, isn't surprising given that we have several orders of magnitude fewer negative samples than are usually reported in competing detectors).

As an (admittedly weak) baseline, we trained a simple, non-boosted, non-tuned classifier based on HOG features. It comes out with a weak AUC of 0.59 suggesting that boosting

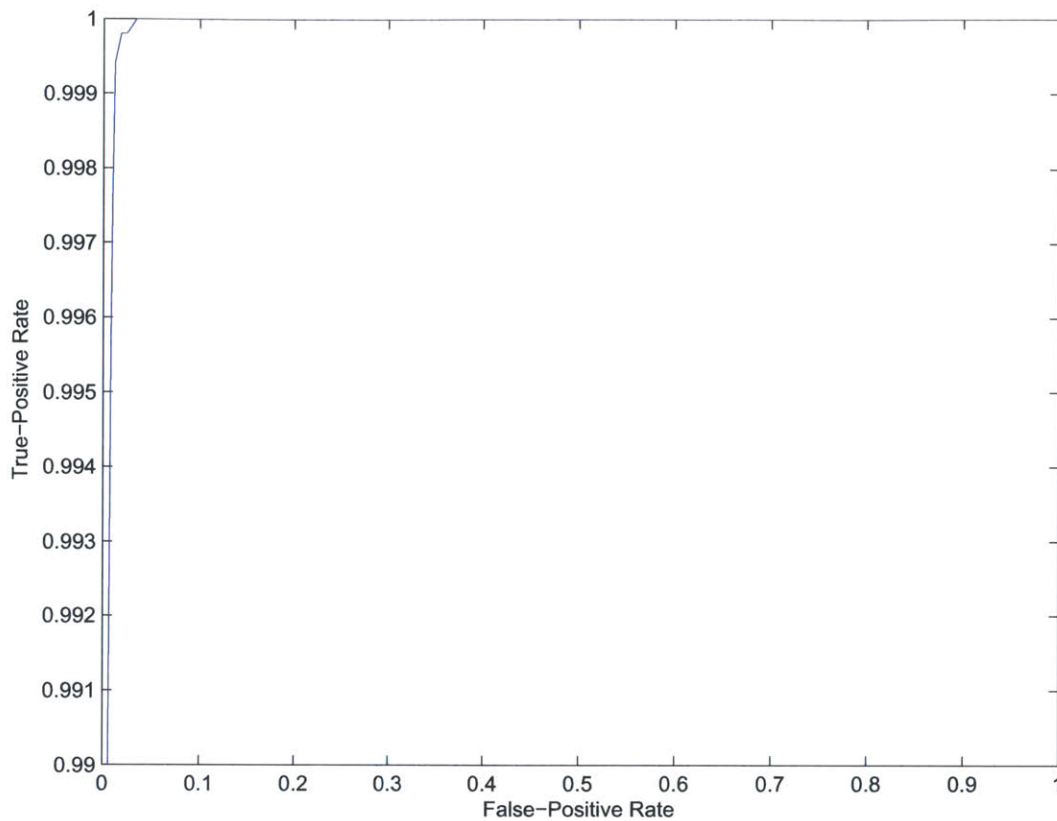


Figure 3-2: ROC Curve of our final face detector; $AUC = 0.998$; in particular, we get 0.7 TPR on the test set with only $1e - 3$ FPR. Accepting slightly more false-positives quickly drives the TPR towards 1.

could certainly help. Some additional explanation of this poor performance is warranted, however, as the HOG descriptor has become rather widely accepted in the computer vision community. First, we extracted only a single feature set for each patch, rather than the 361 SURF feature sets per patch that we were able to choose from in our AUC Boost algorithm. Because there was no boosting and no multiple feature sets to choose from, and the parameters were not finely tuned, it ends up not being so surprising that overall performance is weak.

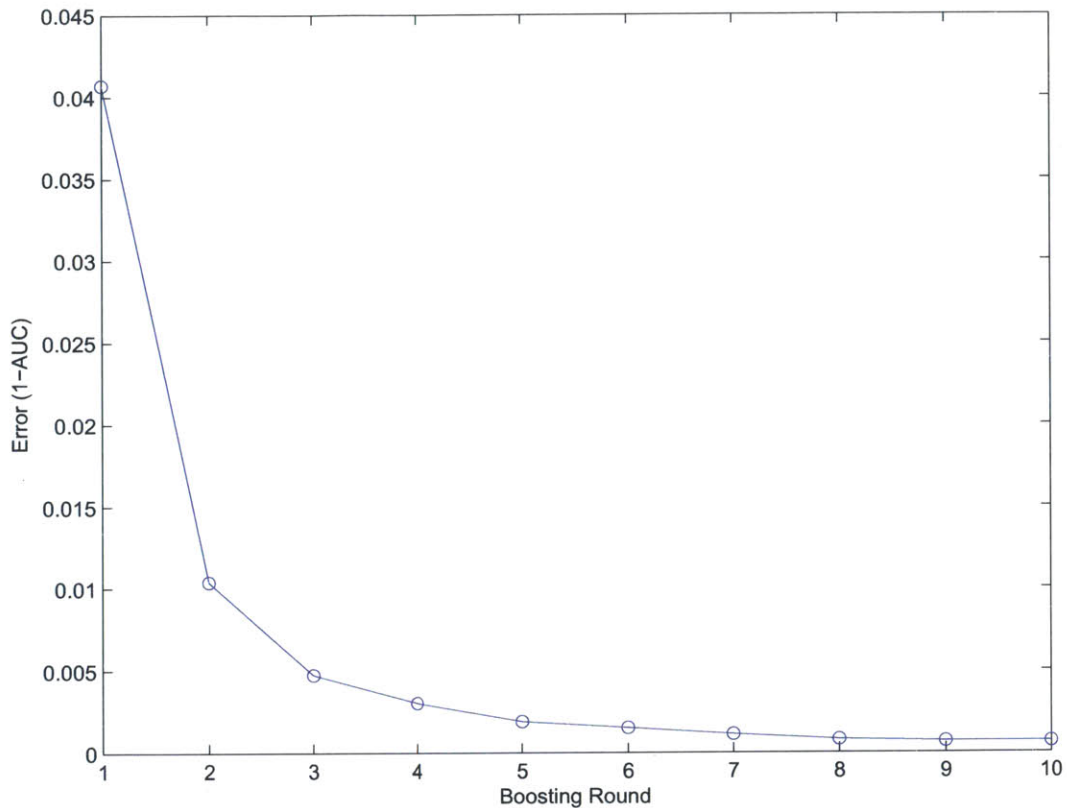


Figure 3-3: Training Error by boosting iteration. We note that the error is already extremely low after the first iteration suggesting that, indeed, our features are indeed discriminative

3.1.7 Limitations

Most obviously, our classifier would benefit from more training data. Since our boosted classifier reached 0 false-positives in training within 10 rounds of boosting, it was unable to continue to improve. The SURF features themselves, therefore, proved to be surprisingly discriminative for our training data, though one wonders if this is not in part because that data was so well-curated (i.e. faces were well-cropped).

Unfortunately, adding more data wasn't so easy due to the way the algorithm was implemented. Specifically, we set out thinking that to speed up development time, we'd extract all features offline and because there are 'only' 80,000 samples $(w, h) = (64, 64)$ patches, that all possible SURF descriptors could be stored in 7GB of RAM (though, it turns out this work on a laptop with 8GB). All this to say that the Matlab code was written with the

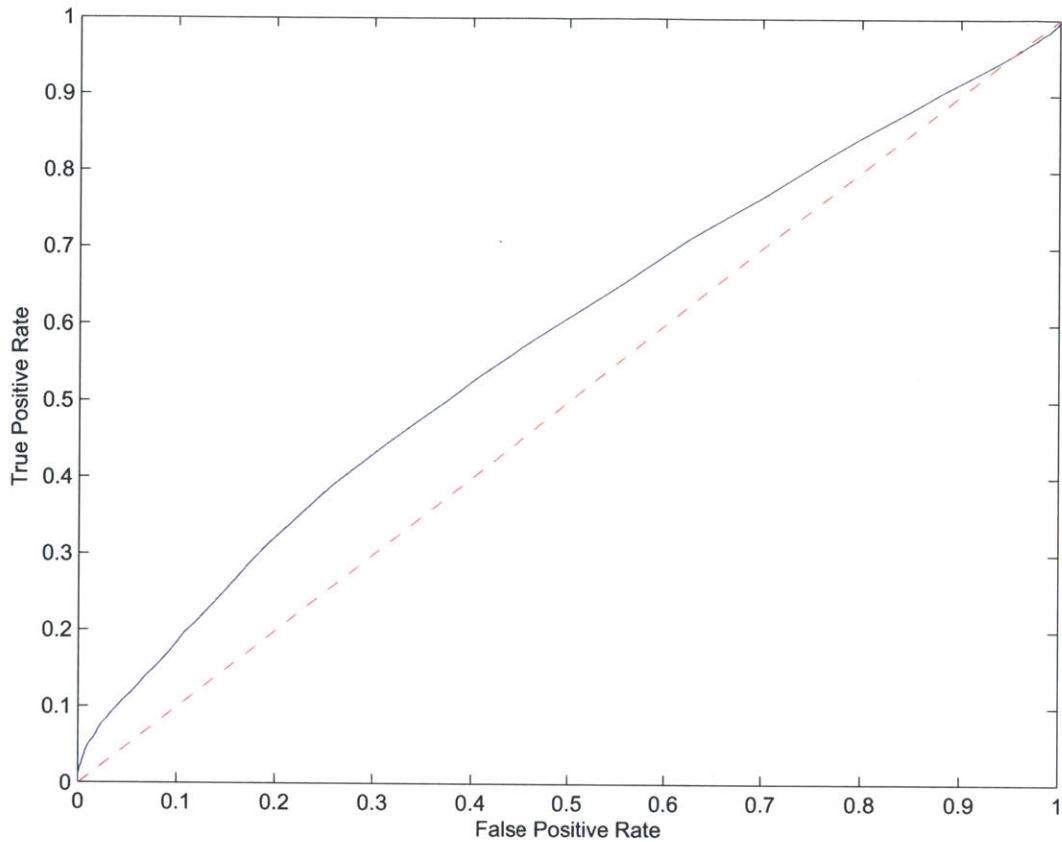


Figure 3-4: ROC Curve of a non-boosted, non-tuned HOG-based classifier; AUC is a weak 0.59 suggesting that further tuning and boosting would likely improve performance.

assumption of all features already being extracted and available, rather than, say, extracting features as needed (which would be necessary for incorporating the necessary billions of examples).

A departure from the cited work was that, although AUC boosting was itself implemented, the cascading framework, in which boosted classifiers are combined, was not implemented because boosting was itself sufficient. The difference, then, between our implementation and that detailed in Figure 3-1, is that in order to benefit from all necessary training data, we augmented the ‘active working set’ maintenance step (iv.) to instead choose as many incorrectly classified examples as it can (as in (e)).

Finally, we note that all elements of this work (except logistic regression, which is built into Matlab) were implemented from scratch. This probably wasn’t ideal from the perspec-

tive of ‘work accomplished’ but our results were surprisingly good, and the process was very instructive. But, in retrospect, a number of implementation details make it so that the code would not immediately scale to the future work needed (more data, principally), and so it would be perhaps at this point be useful to take a closer look at specific implementation choices from others’ implementations.

3.1.8 Summary

We implemented from scratch a boosting framework that permits adaptive FPR by optimizing a classifier’s AUC score. We demonstrated that it performed well on a training set of approx. 70,000 samples, but that its FPR could not be driven lower than $1e - 3$ without substantial erosion of the TPR because there ended up being an insufficient quantity of training samples.

3.2 Social Interaction Detection

Our wearable system detects social interactions so that it may 1) query the wearer about the identities of the people with whom they engaged through non-visual means and 2) so that it ideally could provide distracting notifications (such as by audio) if the wearer is presently engaged.

In this work, we consider a social interaction as the minimum window of time in which the wearer and another person acknowledge each other through both visual and verbal means at least once. Detecting such interactions might be considered to be nearly the same as detecting faces or, alternatively, detecting voice activity. Our experiments (Figure 3-5) show, however, that neither signal is by itself sufficient in practice. People do not always look at each other while interacting (as when they are walking together), nor do they always speak (as when they are simultaneously observing some other event). Furthermore, the wearable assistant is likely to pick up faces and voice activity from people that are nearby, but with whom the wearer is not interacting (as in the cases of a crowded sidewalk where people are transiently passing by, or in the case of a crowded elevator, where people are in proximity but not engaged).

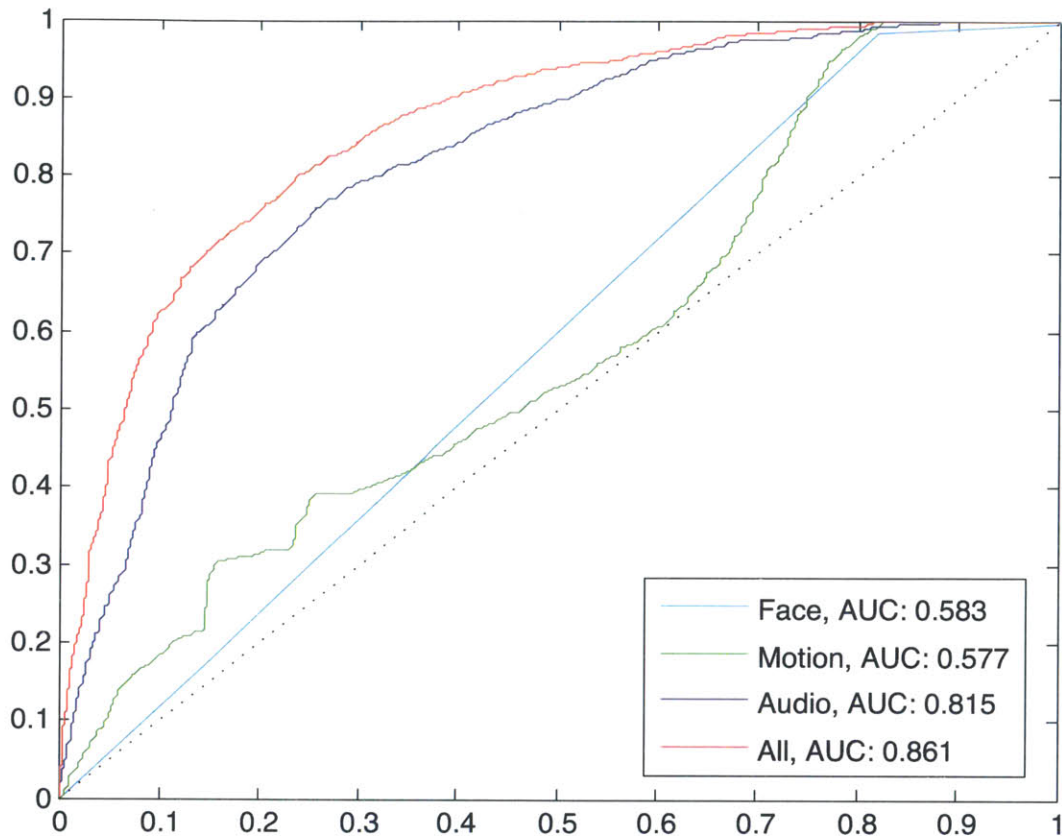


Figure 3-5: ROC Curves of SVM-based detection of social interactions using different groups of features. No single feature grouping (face, motion, audio) performs as well as their combination, which yields an AUC of 0.861.

We thus seek to detect social interactions with a combination of egocentric vision, motion and audio data. To this end, we have built a wearable dataset consisting of images recorded at 1/3 Hz, motion data recorded at 50 Hz, and audio data recorded at 16000 Hz. Images are taken relatively sparsely because the system is designed to last throughout a day; video (and its attendant processing) makes this less feasible.

A dataset was built consisting of 2.5 hours of manually-labeled recordings spread across four separate intervals throughout a single workday. We opted not to record continuously because a typical day includes long stretches of time where rather little is happening (i.e. the wearer is working at a computer). Features are quantized into one-second intervals, of which 76% contain social interactions. Data are randomly split into 70% training and 30%

testing.

Social interaction detection is accomplished by SVM classification of facial features (presence, size, pose) [41] and a subset of the basic energy statistics used in [20], taken on both accelerometer data and the envelope of audio data). Training is iterative and includes multiple rounds of hard-negative mining, in which falsely-classified training data are folded into the training set for a new round of SVM training.

Our results can be seen in Figure 3-5, in which we show ROC curves of classifiers built from facial, audio and motion features. Notably, no single signal performs so well as their combination. We see that a Linear SVM on all features yields AUC of 0.861 and that audio is actually more informative than facial information for detecting social interactions, owing in part due to a low face detection sampling rate and non-ideal camera angle of view (measured from the wearer's chest and equal to 46°).

Because SVM classification can be done relatively cheaply (requiring dot products of the input signal with each support vector), it is reasonable to expect that social interaction detection using our method could be performed online. Additional work would be necessary for this to be reliable, however, as we aren't nearly close enough to reliable detection without significant false positives.

Improving performance of social interaction detection would begin with including a time-aware state-transition model, such as by feeding the above SVM classifications (technically, their distance from the hyperplane) through an HMM [5]. Further improvements could come from the use of more sophisticated audio [6] and accelerometer [26] features.

Chapter 4

Sightless Image Labeling

We seek to provide a (possibly blind) wearer with the ability to teach their wearable system about the acquaintances they'd like it to subsequently identify and inform them about. But this is not so simple as showing them images of the people they interacted throughout the day. Instead, we show work towards providing what is functionally a sightless image labeling interface. By detecting the social interactions that the wearer engages in (as discussed in chapter 3), we can then ask the wearer about the number and identities of the people in each interaction through playback of audio from each interaction.

If each social interaction were considered to only contain a single person and was disjoint in time from other social interactions, then our task would be straightforward: detect each interaction and play audio from each until the wearer can confidently identify the other party.

In reality, however, there will be occasions in which the wearer interacts with multiple people in a single social interaction. This presents challenges in ensuring that the audio-based identification performed by the wearer gets localized to the correct visual representation of the acquaintance – after all, the wearer could have been looking at one person while another was speaking. Or there could be multiple people in-frame, and it not be clear to the system which one is speaking.

We cannot reasonably play the audio of the entire interaction, however, as that could make for a highly cumbersome interface and have significant privacy concerns in that all conversations the wearer engaged in would be recorded. To solve both problems, we instead

	1	2
1	14	0
2	1	13

Table 4.1: Confusion matrix resulting from the use of appearance-based clustering to distinguish between two participants in a social interaction. Columns are predictions, rows are ground truth.

seek to select a small number of short audio snippets from each interaction—enough for identification of each person involved, but not so much as to record the content of the interaction.

In this chapter, we present preliminary results towards this end. Section 4.1.1 discusses Adjusted Mutual Information (AMI), a clustering metric used in a small, appearance-based clustering experiment carried out in Section 4.1. And Section 4.2 discusses strategies for sampling audio for each participant so that user-supplied labels correctly correspond with each person in that interaction.

4.1 Appearance-Based Clustering

We conduct a simple experiment to distinguish the persons involved in a social interaction. From the wearable dataset discussed in Section 3.2 we extract 35 frames containing three separate persons. We then collect HoG features using [9] on extracted faces and cluster them with standard k-means.

We do two experiments. In the first, we attempt to distinguish among two people, and in the second we attempt to distinguish between three. We show results both by comparing the resulting clustering with the ground truth clustering using Adjusted Mutual Information and also by a standard confusion matrix in which class membership of a cluster is determined by majority correct vote.

In experiment 1, we get $AMI = 0.81$, suggesting very good correspondence and, indeed, Table 4.1 shows that only one image is misclassified. In experiment 2, we get $AMI = 0.64$, suggesting still good correspondence, but with greater discrepancies. And we see from Table 4.2 that 5 images are misclassified.

Both experiments are promising, especially given that it seems likely that many social

	1	2	3
1	7	0	0
2	3	10	2
3	0	0	13

Table 4.2: Confusion matrix resulting from the use of appearance-based clustering to distinguish between three participants in a social interaction. Columns are predictions, rows are ground truth.

interactions will be with 1-3 additional people and only more rarely with more. We note that these results assume the number of clusters, i.e. people, is known a priori. In practice, this could be determined by querying the wearer, though ideally it could be determined automatically, perhaps by using varying k in clustering, and evaluating each model with Bayesian Information Criterion [30].

4.1.1 Adjusted Mutual Information

Adjusted Mutual Information [33] is an information-theoretic measure for comparing correspondence between clusterings. We briefly derive it here and discuss and motivate its use compared to other metrics.

Given dataset $S = \{s_1, \dots, s_N\}$ and two clusterings

$$U = \{U_1, \dots, U_R\} \quad \text{and} \quad V = \{V_1, \dots, V_C\}$$

Where $\bigcap_{i=0}^R U_i = \emptyset$ and $\bigcup_{i=1}^R U_i = S$, and similarly for V_i . This just says that the U_i, V_i are a partitioning, or clustering, of S , respectively. Then, the probability that a random data $s \in S$ is contained in some cluster U_i or in cluster V_j is given by the respective equations

$$P_u(i) = \frac{|U_i|}{N} \quad \text{and} \quad P_v(j) = \frac{|V_j|}{N}$$

The probability that s is found in both clusters is given by

$$P(i, j) = \frac{|U_i \cap V_j|}{N}$$

And the mutual information between the two labelings is defined as

$$MI(U, V) = \sum_{i=1}^R \sum_{j=1}^C \frac{P(i, j) \log P(i, j)}{P_u(i)P_v(j)}$$

Mutual information quantifies how much knowing about one clustering tells us about the other. It is symmetric and nonnegative, but is not upper-bounded by a constant and so of limited utility as a general metric for comparing clusterings. Furthermore, [33] demonstrates that mutual information does not take a constant value when comparing random clusterings and tends to grow with the number of clusters. They use a hypergeometric model of randomness to derive an expected value for two random clusterings. This permits a correction similar to the Adjusted Rand Index [15] that ensures random clusterings produce a constant value. This correction yields the adjusted mutual information (AMI):

$$AMI(U, V) = \frac{MI(U, V) - E(MI(U, V))}{\max(H(U), H(V)) - E(MI(U, V))}$$

The entropies of clusterings U, V denote the uncertainty in a data point’s cluster membership:

$$H(U) = - \sum_{i=1}^R P(i) \log P(i) \quad \text{and} \quad H(V) = - \sum_{j=1}^C P(j) \log P(j)$$

The denominator in AMI corrects for randomness and serves as a normalization, as otherwise $MI(U, V) \leq \min(H(U), H(V))$. Furthermore, $AMI(U, V) = 0$ only when equal to its expected value (e.g. that expected by comparing two random clusterings) and $AMI(U, V) = 1$ when clusterings U, V are identical. These bounds make AMI a useful metric for comparing clusterings.

4.2 Label Correspondence and Sub-Sample Selection

Once individuals within a social interaction have been distinguished from one another, it remains to find audio snippets corresponding to each. Presently, our work selects randomly

from within the interaction, irrespective of the presence of speech, but this will only work reliably for interactions in which there is a single participant and when speech is occurring. Ideally, samples would be taken only when speech is detected, perhaps even only the wearer's (for the sake of the privacy of others), but the present implementation is made practical by simply taking extra samples, if needed.

One strategy for handling multiple participants could be to select sequences of images for which the same participant most often remains in frame, the idea being that although we cannot be sure that a person in frame is the same that's currently speaking, we can become increasingly confident with their occurrence in sequence. This would effectively be like considering all social interactions to be with only person, which would inflate the total number of 'social interactions' that the wearer has, particularly when they're actually interacting with more than one person, but regularly shifting attention among the two. And if one person were talking while another was in view then the system would be at risk of misleading the wearer.

Better would be to distinguish participants in a social interaction not just visually through appearance, but also audibly through speaker change detection or speaker segmentation methods such as those surveyed in [19]. We leave this to future work.

4.3 Labeling Interface

An image of our offline labeling interface is shown in Figure 4-1. The day, time and duration of the interact are presented to the wearer through an accessible web interface. Though an image of the person is shown in the interface, it is not needed. Typically, the wearer would listen to the audio clip and then either identify them by typing their name or labeling the interaction as not important. If the audio is not intelligible or does not jog their memory then they can tell the system that they don't know, in which case another audio snippet is retrieved. The number of retries is limited in an attempt to observe the privacy considerations of those that the wearer interacts with.

If social interaction detection were to be continued to be performed offline then additional work could be done to determine the appropriate length of the audio snippets—shorter

is better if privacy is being considered, but they cannot be so short that the wearer cannot properly identify each acquaintance. Our system currently uses five seconds. The system could also bring individuals that the wearer has repeatedly chosen for it not to learn, suggesting that perhaps they really should, in fact, permit the system to identify them.

If social interaction detection were instead to be performed online then audio snippets might need to be played back to the wearer only when they interacted with more than one party. Perhaps the best course in this case would be to adopt a hybrid interface – use the offline interface for interactions with more than two people, and the online interface otherwise.

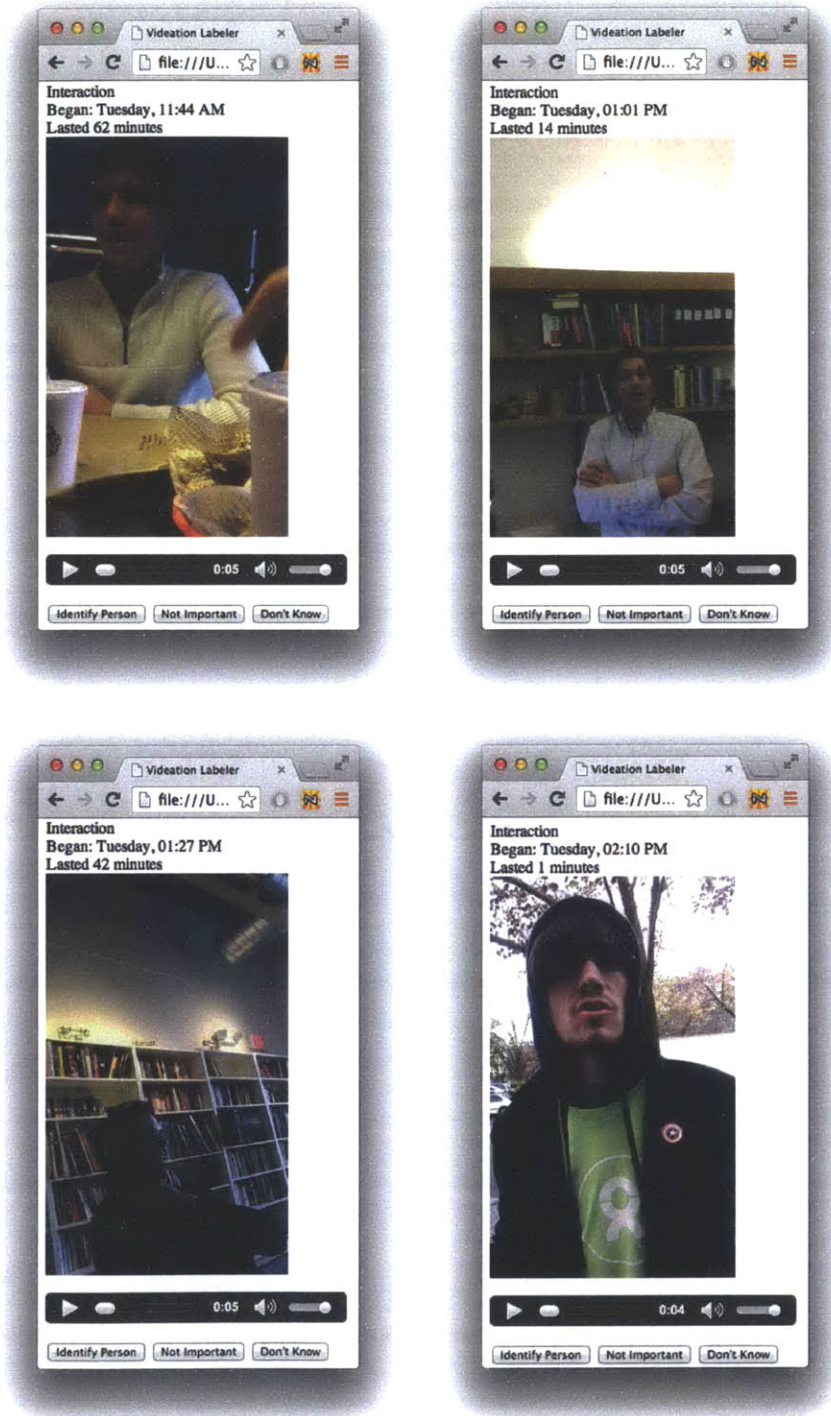


Figure 4-1: Images of the blind-accessible labeling interface, in which each social interaction is summarized, and the wearer is able to provide labels for the people they interacted with by listening to audio clips.

Chapter 5

Conclusions

We have shown an end-to-end implementation of a wearable system that identifies, learns and discreetly communicates the presence of proximate acquaintances to its (possibly blind) wearer. In conducting this work, several research directions of interest emerged, notably the utility of detecting social interactions in egocentric video, audio and motion data and methods for communicating information to a person engaged in other activities, particularly social interaction. While much of the work is imperfect, it nevertheless demonstrates the viability of such a system. We conclude with discussion of softer considerations that were at least considered in the construction of our wearable system, and that would need to be revisited in making it broadly available and touch upon future work.

5.1 Utility, Privacy and Physical Comfort

One challenge facing wearable computers is that the value they provide must be high, or else the inconvenience of wearing them must be very low. It will be particularly difficult to find people willing to adopt wearables if they are physically cumbersome, aesthetically distasteful or if they encroach upon social norms. And just as there are only certain times and places in which it is currently acceptable to wear headphones or Bluetooth headsets, so too is it likely that there may only be specific contexts under which wearables such as this work are acceptable to be used in.

Particularly challenging with wearables that involve a camera is the fear from others

that they might be getting recorded. As evidenced by some of the backlash over Google Glass, the presence of a camera appears to be enough to set this concern off, even if the wearer assures that it is not, in fact, recording. Opinions appear to range from the notion that people will become increasingly comfortable with the potential that they are being recorded constantly to suggesting that either markets or governments will force wearables to not have the ability to discreetly record image or audio data.

An interesting angle that may work towards easing adoption of this work, if not wearables more generally, is the highlighting of their immediate applications for individuals with disabilities. A jacket or pendant that identifies proximate acquaintances is just one of many potential applications here.

5.2 Future Work

Our wearable system currently identifies proximate acquaintances from facial recognition alone, and social interaction detection and acquaintance learning are both performed offline. A future implementation would change being that the Acquaintance Model of Figure 2-1 to includes recognition not just on faces, but also on visual attributes of the body (including clothing, skin color and hairstyle), as well as audio attributes of voice and contextual attributes (including location, wearer calendar). Furthermore, Interaction Detection would be performed online so that the system can both modify how it informs the user of nearby acquaintances (based on whether they're already engaged with others or not), and also so that it can query the user about the identities of people they interact with at an opportune time shortly after the interaction itself took place.

Analysis of social interactions would also be improved so that the number of participants could be determined automatically and so that determining correspondences between their voice and appearance incorporated speaker segmentation techniques.

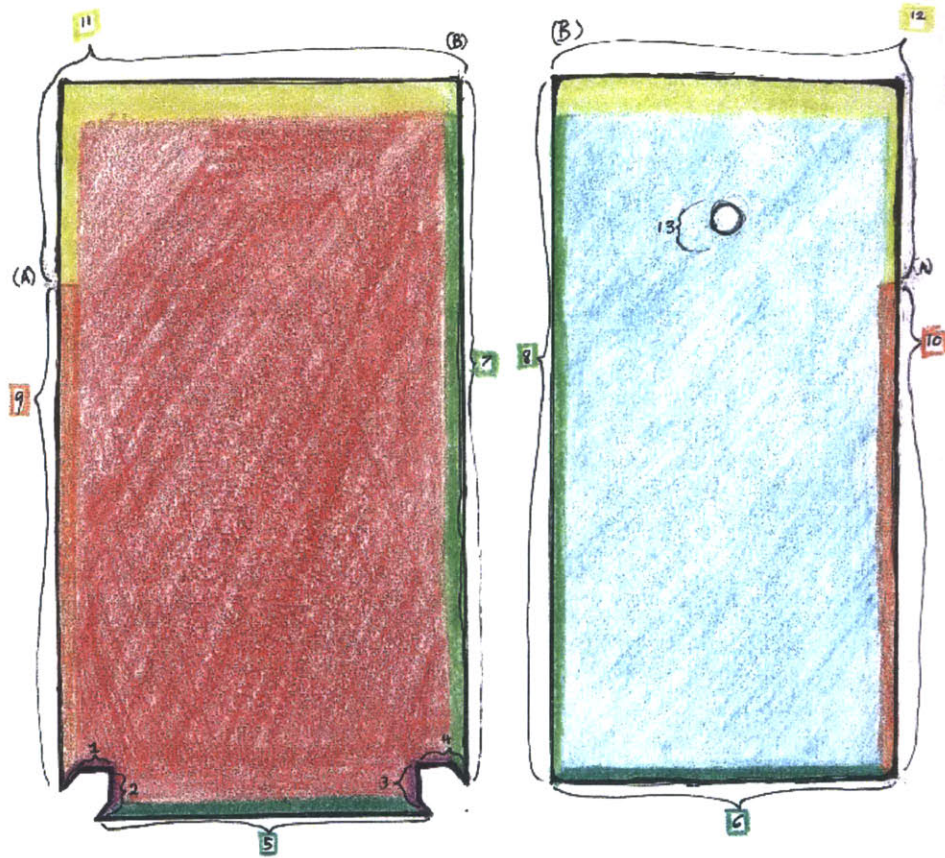
The hardware would be further miniaturized, perhaps into a small pendant, so that the wearer would not be tied to a specific jacket or a large lanyard. The camera would incorporate a much wider angle of view, or be a combination of cameras that together total at least 90°, in which case other techniques for analyzing social interactions would

become more feasible. Greater CPU power can provide greater flexibility to do additional and more complex processing locally, reducing or eliminating the need of a server. And the software platform would be exchanged for one that was more specifically built for wearable applications.

Finally, an interesting direction for this work would be to use the wearable system to measure the number of serendipitous social interactions that the wearer could have engaged in, but for whatever reason, chose not to, or otherwise couldn't. This could, for instance, provide a platform for studying the social interaction patterns of people on a large scale, and would be of particular interest for more quantitatively understanding populations that we believe have fewer or more limited social interactions.

Appendix A

Figures




Actual-size pattern. Fabric should be somewhat stretchy, but not too thin, lightweight, or excessively stretchy.

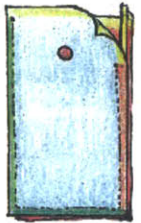
I used cotton jersey knit from a T-shirt; this worked very well.


You will need a serger, not just a sewing machine. For the serging, I'd recommend woolly nylon thread because it can stretch with the fabric.

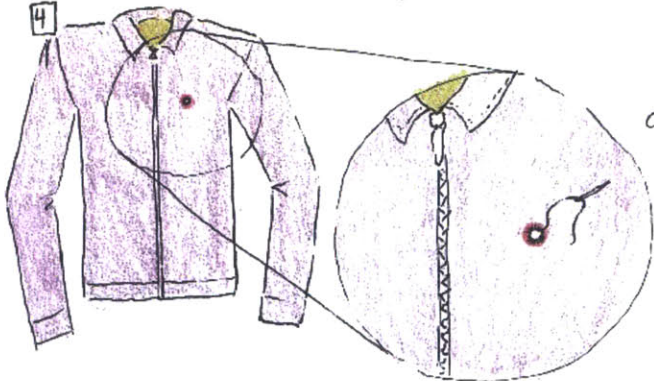
You will need a zipper at least 6" long.


Figure A-1: Materials and pattern specifications for the Jacket component of our wearable system, in which a wearable camera is discreetly placed within a pocket sewn into its inner-lining, with its lens peering out through a small hole.

1  Serge 1 to 2 and 3 to 4.

2  Serge 5 to 6, 7 to 8, and 9 to 10.
Optional: using a short stitch length, machine sew all the way along this seam to reinforce it.

3 Sew zipper between 11 and 12. It is important that the zipper zip from the side (A) to the top (B), because it is much easier to close that way than the other direction. I usually handsew the zipper in. 

4  Cut hole at appropriate location on the front of the jacket, from which the camera could get a good head-and-torso view of another person from long-range all the way to conversation distance. Hand-stitch around hole for a clean opening.

5 With phone in pocket, line up the pocket on the inside of the jacket so that the holes match up, and around the opening, handsew the jacket to the pocket. 


6 Turn jacket inside out. Taking care that the pocket hole stays comfortably aligned with the camera lens, pin the perimeter of the pocket to the lining (do NOT pierce all the way through to the outside of the jacket) and sew the pocket to the lining (once again, just to the lining, not to the outer fabric of the jacket).  Your videaton pocket is now complete! You may also choose to sew in additional inner pockets to hold extra batteries. These pockets can be much simpler and use snaps instead of zippers.

Figure A-2: Steps for creating the Jacket component of our wearable system in which a wearable camera is discreetly placed within a pocket sewn into its inner-lining, with its lens peering out through a small hole.



Figure A-3: Images of people using different configurations of wearable cameras, shown to the public, and asked whether they felt the person in each image would look 'normal,' 'peculiar,' or 'weird' if they encountered them in a super market while shopping for groceries. In order are the control (no camera), ear camera, chest camera, pinwheel (for comparison), head camera and face camera.

Bibliography

- [1] Rekognition: Integrated visual recognition solution based in cloud. <https://rekognition.com/>. Accessed: 2013-06-01.
- [2] Recommendation m.1677-1: International morse code. International Telecommunication Union, October 2009.
- [3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.
- [4] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, 1997.
- [5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.
- [6] Ekapol Chuangsuwanich and James R Glass. Robust voice activity detector for real world applications using harmonicity and modulation frequency. In *INTERSPEECH*, pages 2645–2648, 2011.
- [7] James Clawson, Nirmal Patel, and Thad Starner. Digital kick in the shin: On-body communication tools for couples trapped in face-to-face group conversations. *Workshop on Ensembles of On-Body Devices, MobileHCI10*, 20, 2010.
- [8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
- [9] Piotr Dollár. Piotr’s Image and Video Matlab Toolbox (PMT). <http://vision.ucsd.edu/pdollar/toolbox/doc/index.html>.
- [10] A. Fathi, J. K. Hodgins, and J. M. Rehg. Social interactions: A first-person perspective. In *CVPR*. IEEE, 2012.
- [11] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

- [12] I. Fette and A. Melnikov. The WebSocket Protocol. Number 6455 in Request for Comments. Internet Engineering Task Force, December 2011.
- [13] Albert S Huang, Edwin Olson, and David C Moore. Lcm: Lightweight communications and marshalling. In *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*, pages 4057–4062. IEEE, 2010.
- [14] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.
- [15] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.
- [16] TM Huurre and HM Aro. Psychosocial development among adolescents with visual impairment. *European Child & Adolescent Psychiatry*, 7(2):73–78, 1998.
- [17] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.
- [18] NJ Kemp. Social psychological aspects of blindness: A review. *Current Psychological Reviews*, 1(1):69–89, 1981.
- [19] Margarita Kotti, Vassiliki Moschou, and Constantine Kotropoulos. Speaker segmentation and clustering. *Signal processing*, 88(5):1091–1124, 2008.
- [20] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2), 2011.
- [21] J. Li, T. Wang, and Y. Zhang. Face detection using surf cascade. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2183–2190. IEEE, 2011.
- [22] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [23] O. Ludwig, D. Delgado, V. Gonçalves, and U. Nunes. Trainable classifier-fusion schemes: an application to pedestrian detection. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pages 1–6. IEEE, 2009.
- [24] Walterio W Mayol-Cuevas, Ben J Tordoff, and David W Murray. On the choice and placement of wearable vision sensors. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(2):414–425, 2009.
- [25] Anja Moos and Jürgen Trouvain. Comprehension of ultra-fast speech–blind vs. 'normally hearing' persons. In *Proceedings of the 16th International Congress of Phonetic Sciences*, pages 677–680, 2007.

- [26] Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. Online pose classification and walking speed estimation using handheld devices. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 113–122. ACM, 2012.
- [27] Vignesh Ramanathan, Bangpeng Yao, and Li Fei-Fei. Social Role Discovery in Human Events. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.
- [28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.
- [29] C. Sanderson and B. Lovell. Multi-region probabilistic histograms for robust and scalable identity inference. *Advances in Biometrics*, pages 199–208, 2009.
- [30] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.
- [31] Steve Tauroza and Desmond Allison. Speech rates in british english. *Applied Linguistics*, 11(1):90–105, 1990.
- [32] Vincent B Van Hasselt. Social adaptation in the blind. *Clinical Psychology Review*, 3(1):87–102, 1983.
- [33] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080. ACM, 2009.
- [34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.
- [35] Bruce N Walker, Raymond M Stanley, Nandini Iyer, Brian D Simpson, and Douglas S Brungart. Evaluation of bone-conduction headsets for use in multitalker communication environments. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 1615–1619. SAGE Publications, 2005.
- [36] James R Williams. Guidelines for the use of multimedia in instruction. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 42, pages 1447–1451. SAGE Publications, 1998.
- [37] Piotr A. Wozniak. *Economics of Learning: New aspects in designing modern computer aided self-instruction systems*. PhD thesis, University of Economics in Wrocław, 1995.

- [38] R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [39] Lucy Yardley, Lisa McDermott, Stephanie Pisarski, Brad Duchaine, and Ken Nakayama. Psychosocial consequences of developmental prosopagnosia: A problem of recognition. *Journal of Psychosomatic Research*, 65(5):445–451, 2008.
- [40] Zhefan Ye, Yin Li, Alireza Fathi, Yi Han, Agata Rozga, Gregory D. Abowd, and James M. Rehg. Detecting eye contact using wearable eye-tracking glasses. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, 2012.
- [41] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*. IEEE, 2012.