MIThril 2003: Applications and Architecture

Rich DeVaul, Michael Sung, Jonathan Gips, Alex "Sandy" Pentland
Media Laboratory, Massachusetts Institute of Technology
{rich,msung,jgips,sandy}@media.mit.edu

Abstract

In this paper we describe the MIThril 2003 wearable computing research platform. MIThril 2003 is a proven, accessible architecture that combines inexpensive, commodity hardware, a flexible sensor/peripheral interconnection bus, and a powerful, light-weight distributed sensing, classification, and inter-process communications software layer to facilitate the development of distributed real-time multimodal and context-aware applications.

MIThril 2003 extends the previous MIThril modular architecture into the domain of large-scale wireless group applications by leveraging the availability of inexpensive Linux-based PDA hardware combined with innovative open-source software and custom sensor hardware. We demonstrate the power and functionality of MIThril 2003 by describing compelling real-world wearable research applications created using MIThril 2003 technology.

1. Introduction

The MIThril project [6] started as an attempt to remedy the substantial human factors, flexibility, and robustness problems plaguing wearable computing research at the end of the 1990s. From these beginnings, MIThril evolved towards a practical, modular system of hardware and software for research in wearable sensing and context-aware interaction [1, 20].

In 2000, the defining feature of MIThril was the modular, distributed, clothing-integrated design based on a unified power/data bus, allowing us to put sensing, computing, and interaction resources where they were most useful and appropriate [6, 16].

In 2003, the advent of inexpensive wireless-capable Linux-PDA hardware allows us to redefine MIThril as a multi-user wireless distributed wearable computing environment, supporting dozens of interacting users and large-scale interaction and sensing experiments.

This paper focuses on three important components of the MIThril 2003 architecture: a PDA-centric MIThril hardware configuration, the Enchantment Whiteboard IPC system, and the Enchantment Signal real-time machine learning infrastructure.

2. Motivation

The development of the original MIThril 2000 platform grew from our need for a flexible, reliable, modular prototyping system for the development of unobtrusive wearable research applications. Off-the-shelf hardware options were limited to repackaged notebook computers, heavy-duty industrial wearables such as those provided by Xybernaut, or PC104-based systems like the original MIT Lizzy [25]. Our work with these systems led us to understand the importance and need for a modular, flexible sensing and interaction system that was light-weight, reconfigurable, and easy to extend, use, and maintain. We also wanted to create an accessible system so that others could easily replicate our work.

MIThril 2000 succeeded in some ways and failed in others. We were successful in producing a very flexible research platform that largely met our technical and ergonomic requirements. Our choice of Linux as the foundation OS and I2C as the primary body-bus protocol proved to be quite successful. Likewise, the modular architecture made troubleshooting and reconfiguration relatively easy. It also allowed the platform to evolve piece-wise without the need for comprehensive reengineering. However, the relatively large number of components, high level of hardware expertise required for fabrication, and expense (~$3k in component costs in 2001) made it difficult to build and maintain more than half a dozen units.

Beginning in 2001 we saw the need for a light-weight data acquisition system that could talk to MIThril sensors, primarily for biomedical and social networks research applications. Collaborating with Vadim Gerasimov, we developed the Hoarder system (described below) to meet these needs. For the first time, we were able to run experiments with dozens of users. However, the interaction, communications, and signal processing capabilities of these systems were extremely limited.

The breakthrough came with the availability of inexpensive, Linux-capable PDAs with significant signal processing and communications capabilities. By combining such devices with the Hoarder and MIThril body bus peripherals, we created a wearable with many of the capabilities of the original MIThril system, but at a fraction of the original MIThril 2000 system complexity

and cost. Such systems do not provide the processing power of a Xybernaut Mobile Assistant or Charmit and are not suitable for some highly computationally intensive tasks on-body such as face recognition. However, they do provide sufficient processing power for a range of interesting sensing, classification, and user-interaction applications, examples of which are described at the end of this paper.

Good hardware is not useful without correspondingly good software. MIThril's Linux foundation provides excellent functionality and support for a wide range of software packages. However, we found critical problems in distributed inter-process communication, signal processing, and sensor data classification that were not addressed by either the operating system or currently available research tools.

To address these problems we created an architecture to combine the best features and practices from a range of research systems and methodologies, and to do so in an open, modular, and flexible way. In this paper we describe three important software systems that form the foundation of this architecture: The Enchantment Whiteboard, Signal System, and Real-Time Context Engine. These tools address critical needs in the development of wearable research applications while imposing minimal constraints on the nature of these applications.

3. MIThril 2003 Hardware Overview

The MIThril 2003 hardware architecture is a highly flexible, modular system that is tied together by wired/wireless networking protocols and a unified multi-protocol wired power/data bus, called the MIThril Body Bus, for sensors and peripherals.

Two related MIThril 2003 hardware configurations are discussed in this paper. The first is a PDA-centric system with sensor hub and distributed sensor network, and the second is a stand-alone microcontroller-based data acquisition system with sensor network. An example of a PDA configuration is shown in Figure 1.

3.1 MIThril 2003 Computing Nodes

For applications requiring real-time data analysis, peer-to-peer wireless networking, full-duplex audio, or graphical interaction, MIThril 2003 employs a Linux-based PDA, such as the Sharp Zaurus SL-5500 shown in Figure 1. We chose the Zaurus SL-5500 because it provides excellent Linux support combined with a range of capabilities in a small, inexpensive package.

The Zaurus SL-5500 is a complete embedded Linux system. It provides a 206-Mhz StrongARM processor, 64 MB SDRAM, 16 MB Flash, CF and SD expansion slots, full duplex audio, qVGA color touch screen, and an integrated QWERTY keyboard. The StrongARM provides sufficient computing power for moderate bandwidth signal processing and real-time classification.

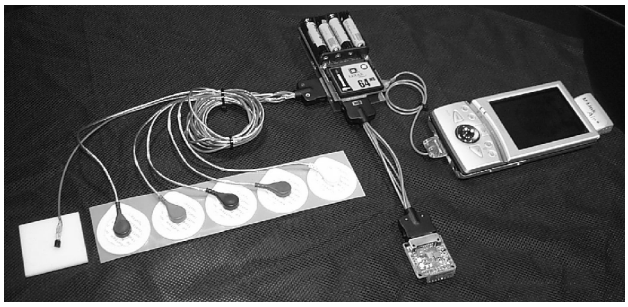

Figure 1: MIThril 2003 system, comprised of the Zaurus PDA (right) with Hoarder sensor hub and physiological sensing board (top), EKG/EMG/GSR/ temperature electrodes and sensors (left) and combined three-axis accelerometer, IR tag reader, and RS232/I2C bridge (bottom)

The CF card slot enables a rich variety of peripherals to be attached, including cell-phone modems, image and video cameras, Bluetooth and 802.11b (WiFi) wireless, and even head mounted displays – we are working on a Linux driver for the CF MicroOptical VGA color HMD produced by Interactive Imaging Systems. The Zaurus also provides an RS232 serial port, which we use to interface with the MIThril sensor hub described below.

3.2 MIThril 2003 Sensor Hubs

A sensor hub is used to interface the Zaurus SL-5500 with the MIThril body bus. This hub bridges between the RS232 serial interface on the Zaurus and the Phillips I2C multi-device serial protocol used on the MIThril body bus.

When connected to the Zaurus, the sensor hub is responsible for sensor data acquisition, buffering, and sequencing. We are currently using the Hoarder board [10] as the sensor hub, which also provides stand-alone data acquisition capabilities in the form of a CF storage card interface, a real-time clock for sequencing, battery power, and an optional FM wireless transceiver. The Hoarder supports sensor daughter boards as well as I2C devices through its MIThril body bus.

The Hoarder can also be used without the Zaurus as a stand-alone data acquisition system. This is particularly useful for large-group applications that do not require real-time processing, WiFi wireless or complex user interaction. We have used this configuration in social network experiments [3], real-time critical health

monitoring [27], and identifying activities of daily living [12].

A smaller, multi-function sensor/hub device was recently designed that provides many of the same capabilities as the Hoarder, but in a smaller package without support for stand-alone data acquisition. We are now in the process of designing the next-generation data acquisition/sensor hub, shrinking and integrating it into a wearable docking cradle for the Zaurus. In addition to being smaller and better integrated with the Zaurus, the new sensor hub will provide regulated power for the Zaurus as well as the MIThril bus and low-power addressable megabit peer-to-peer wireless (2.4 GHz) using the Nordic VLSI nRF24xx parts.

3.3 MIThril 2003 Sensor Hardware

MIThril 2003 sensors are either stand-alone microcontroller-based devices that speak I2C on the MIThril body bus, or analog/digital sensor daughter boards for the Hoarder sensor hub. Currently available stand-alone sensor designs include accelerometers, IR active tag readers, battery monitors, and a GPS unit. Any number of these sensors can be combined through the use of the MIThril body bus and passive junctions.

Two major types of Hoarder daughter boards are currently in use: a multi-sensor board combining a digital tri-axial accelerometer and IR tag reader sensor with an analog microphone circuit, and a physiological sensing board providing analog 2-channel EKG/EMG, 2-channel galvanic skin response (GSR), and skin temperature sensors.

The analog ports of the Hoarder also allow us to interface with a wide range of commercially available sensors, including pulse oximetry, respiration, blood pressure, EEG, blood sugar, humidity, and CO_2 sensors. RS232 interfaces provided by the Zaurus, Hoarder, and the new multi-sensor hub also provide pathways to interface with standardized serial-interface sensor packages and interface boards.

3.4 MIThril 2003 Networking

The MIThril 2003 PDA configuration supports wireless IP networking through the CF interface using the 802.11b wireless protocol. This low-cost wireless networking capability is a crucial enabling feature, allowing us to implement multi-node, distributed wearable applications.

The 802.11b protocol provides sufficient bandwidth for many systems to simultaneously stream full-duplex audio and sensor data, as well as sending lower-bandwidth data such as text messaging and state information. With wireless connectivity, data can be streamed to off-body resources for a variety of purposes, including data logging and storage, visualization/display, and signal processing. Persistent network servers are also used for directory services and resource discovery.

4. MIThril 2003 Software

This section describes three important parts of the MIThril 2003 software infrastructure: the Enchantment Whiteboard, the Enchantment Signal system, and the MIThril Real-Time Context Engine. These tools provide the foundation for developing modular, distributed, context-aware wearable and ubiquitous computing applications. Please see ***http://www.media.mit.edu/wearables/mithril/enchantment*** and ***http://www.media.mit.edu/wearables/mithril/context*** for more information and to download software packages.

4.1 The Enchantment Whiteboard

The Enchantment Whiteboard system is closely related to blackboard and whiteboard systems that have developed in artificial intelligence research over the last twenty-five years [15, 21, 22]. Unlike traditional inter-process communications systems (such as RMI, Unix/BSD sockets, etc.) which are based on point-to-point communications, the Enchantment Whiteboard is based on a client/server model in which clients post and read structured information on a whiteboard server. This allows any client to exchange information with any other client without the attendant (n^2) complexity in negotiating direct client-to-client communication; in fact, this is possible without knowing anything at all about the other clients.

This does for inter-process communication what web browsers and web servers do for document publishing – it provides a uniform structure and systematic organization for the exchange of information that does not require synchronous communications.

The Enchantment Whiteboard goes beyond the web server analogy by allowing clients to subscribe to portions of the whiteboard, automatically receiving updates when changes occur. It allows clients to lock a portion of the whiteboard so that only the locking client can post updates. And it even supports symbolic links across servers, allowing whiteboards to transparently refer to other whiteboards across a network. The Enchantment Whiteboard is also lightweight and fast, imposing little overhead on the communications.

The Enchantment Whiteboard is intended to act as a streaming database, capturing the current state of some system (or person, or group) and on modest embedded hardware can support many simultaneous clients distributed across a network and hundreds of updates a second. We have even demonstrated the ability to use the

Enchantment Whiteboard with the Signal system for bandwidth-intensive VoIP-style audio communications.

The Whiteboard library presents an interface with the following core functionality: publishing, retrieving, subscribing, and locking. When data is retrieved from the Whiteboard, it is not generally removed and remains for other applications to query. According to the client's specification, data that is published to the Whiteboard can either be persistent or transient.

As an example to illustrate the interaction of a distributed application using the Enchantment Whiteboard System, in Figure 2 we diagram an instant messaging system called Echat, which was prototyped in less than an hour using the Enchantment API and shell scripts.

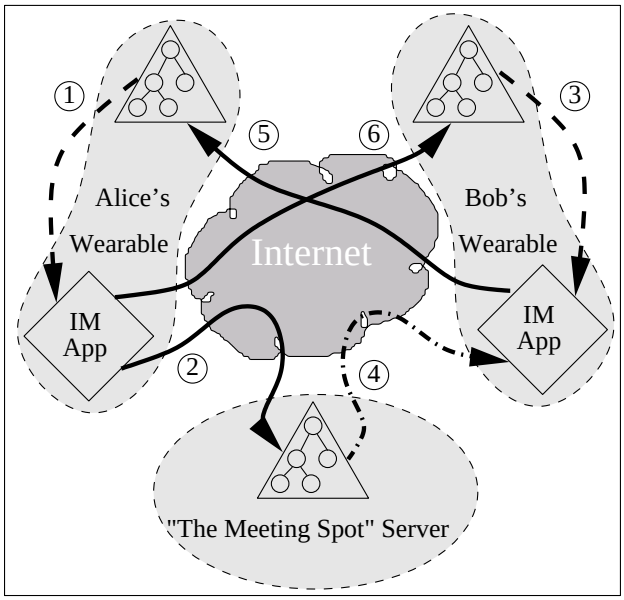

Figure 2: Whiteboard Example: Echat

Figure 2 shows the Instant messaging (IM) application with lines indicating Enchantment IPC Whiteboard calls, line style denoting call type, and arrows indicating flow of information. (1) Alice's IM app subscribes to updates from her wearable's context server and (2) posts a link to her context server on "The Meeting Spot" context server. (3) Bob's IM app subscribes to his own context server and (4) checks "The Meeting Spot" server where he retrieves Alice's context server information. If Bob wants to message Alice, (5) he publishes his message along with his context server address to her context server. (6) Alice can then respond in kind.

The system is considered a whiteboard because multiple processes operate in parallel on the servers' hierarchical data. This differs from blackboard systems where, typically, a blackboard manager directs control to knowledge source modules that in turn post data to the central repository [23].

The Enchantment Whiteboard system also shares many properties of coordination languages such as Linda [2], a tuple space language, and its successors, which include LIME [21], TSpaces [12], and JavaSpaces [26]. Both the Enchantment Whiteboard and Linda systems offer many of the same basic primitives to processes such as the ability to publish to and read from globally addressable spaces that contain name-value tuples. The Enchantment Whiteboard system differs from coordination languages in its use of symbolic links instead of replication and its emphasis on hierarchically structured data, which we believe is well suited to context-aware applications, in place of tuples.

4.2 The Enchantment Signal System

For higher bandwidth signals, especially those related to the sharing and processing of sensor data for context aware applications, we developed the Enchantment Signal system. The Signal system is intended to facilitate the efficient distribution and processing of digital signals in a network-transparent manner. The Signal system is based on point-to-point communications between clients, with signal "handles" being posted on Whiteboards to facilitate discovery and connection. In the spirit of Whiteboard interactions, the Signal API abstracts away any need for signal produces to know who, how many, or even if, there are any connected signal consumers.

Any type of structured numeric data can be encoded as a signal. Signal producers may be sensors, feature extractors, filters, or regression systems, where many producers are also consumers. A typical organization is a sensor signal producer talking to a feature extraction signal consumer, which in turn produces a feature signal that is consumed by one or more modeling or regression systems.

The results of modeling or regression can themselves be signals, or (more typically) posted on a whiteboard for other clients to use. This model allows for the implementation of principled statistical machine learning systems, as described below in Section 4.3.

Since Enchantment runs on top of non-real-time Linux kernels and employs a multi-hop, distributed organization, latencies are additive. Thus, without the use of predictive algorithms, the system can only be as real-time as the latencies that are present in the Enchantment subsystems. Despite this limitation, accurate context classification can be made by time stamping all signals at their source. Also in general, we find that the exhibited latencies are small enough such that real-time responsiveness in user interaction is supported by most applications.

4.3 MIThril Real-Time Context Engine

The MIThril Real-Time Context Engine [7] is an open-source, light-weight, and modular architecture for the development and implementation of real-time context classifiers for wearable applications. The high-level organization of the Context Engine is based on the classification process described by Duda and Hart, a standard reference [8]. It is implemented using the Enchantment Signal system and Kevin Murphy's Bayes Net Tool Kit [19], a freely available, powerful graphical model and Bayesian inference system for Matlab.

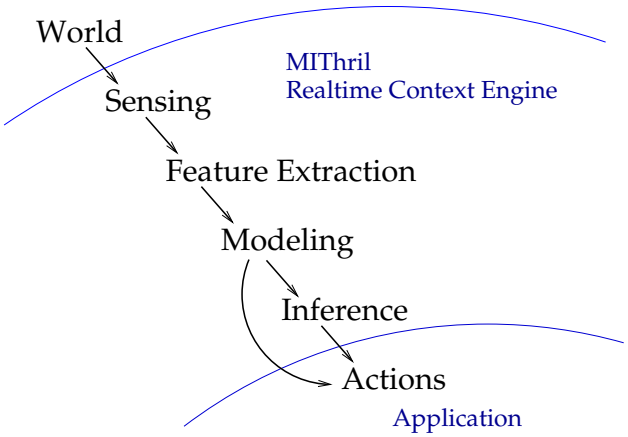

Figure 3: MIThril Real-Time Context Engine

The MIThril Real-Time Context Engine abstracts the process of sensing, modeling, and inference as a four-stage process, as shown in Figure 3 above. The stages are as follows:

1. **Sensing:** In the sensing stage, a digital sensing device measures something in the real (analog) world, resulting in a digital signal of sampled values. For example, a microphone sensor converts continuous fluctuations in air pressure (sound) into discrete sampled values with a specified resolution, encoding, and sampling rate.

2. **Feature Extraction:** In the feature extraction stage, a raw sensor signal is transformed into a feature signal more suitable for a particular modeling task. For example, the feature extraction stage for a speaker identification classification task might involve converting a sound signal into a power spectrum feature signal.

3. **Modeling:** In the modeling stage, a statistical model (such as a Gaussian mixture model, Hidden Markov Model, or Support Vector Machine hyper-plane classifier, etc.) is used to classify a feature signal in real time. For example, a Gaussian mixture model could be used to classify accelerometer spectral features as walking, running, sitting, etc.

4. **Inference:** (Matlab implementation only) In the inference stage, the results of the modeling stage, possibly combined with other information, are fed into a Bayesian inference system for complex interpretation and decision-making.

The first three stages of this four-stage process have been implemented in C and C++ using the Enchantment Signal and Whiteboard system. (We have not yet had a practical need for a real-time inference stage).

It is important to note that the Real-Time Context Engine is an extensible modeling framework, not an instance of a particular model. So far we have only had need for relatively simple statistical models, but more complex models (such as Clarkson's two-level HMMs for unsupervised clustering of sensor data [4]) are naturally accommodated by this framework.

This classification process assumes that features and model structure have been chosen, and that the model parameters are known. A full discussion of the model development process is well beyond the scope of this paper; more information may be found in [8, 14].

4.3.1 Example Classifier Implementation As an example demonstrating the interaction between the MIThril Real-Time Context Engine and the Enchantment Whiteboard/Signal infrastructure, we now describe an implementation of a highly reliable, low-latency multi-class activity classifier (*e.g.*, standing, walking, running). A tutorial for developing the three-class activity classifier using Context Engine methodology and a single three-axis accelerometer is available on the web [5] and described in more detail in [7]. We briefly describe this implementation below:

Hardware: a single three-axis accelerometer connected to a wireless-enabled MIThril 2003 system through the Hoarder sensor hub.

Software: Zaurus running a local whiteboard server and several signal/whiteboard clients, as follows:

1. An Enchantment Signal generator samples the accelerometer at 50Hz, resulting in a signal composed of raw 3-axis acceleration vectors. (a signal of integer 3-vectors). A handle to this signal is published on the whiteboard at localhost:/signals/accel.

2. An Enchantment Signal client, a power spectrum feature extractor, subscribes to the raw accelerometer signal at localhost:/signls/accel and computes a short-time windowed power spectrum of the magnitude of the accelerometer signal. It generates its own signal of 31 element feature vectors and publishes a handle on the whiteboard at localhost:/signals/accel _freq.

3. A third Signal client, a Gaussian mixture model classifier, is created. It reads its model parameters from a Matlab-produced configuration file and subscribes to the power spectrum feature published at localhost:/signals/accel_freq. As the model runs, the classification results (a list of labels and probabilities) are published on the whiteboard at localhost:/classifiers/activity.

4. Logging: (Optional) Signal clients running on a desktop server connect to the Zaurus Whiteboard server and signal producers through the wireless network. These desktop clients log the signals and classifier output for later analysis.

5. Visualization: (Optional) For demo purposes, the output of the accelerometer signal generator, the feature extractor, and Gaussian classifier are visualized simultaneously using X Windows-based visualization clients running on a laptop.

Figure 4: MIThril 2003 real-time classification demo.

Figure 4 shows a demo of the complete activity classification system in action. Accelerometer data is being classified in real-time, and the classification results and EKG/GSR/temperature are wirelessly streamed to a remote computer with a projector

In addition to accelerometer-based activity classifiers, we have implemented an audio-based whistle classifier (it classifies whistled notes, distinguishing them from speech, silence, and each other) and an electromyogram (EMG) classifier that is capable of discriminating between different gestures based on muscular activity.

At present the main limitation of the Real-Time Context Engine is the small number of model types for which we have real-time support; although we can train a wide variety of models under Matlab, at present we have only implemented working Gaussian /Markov classifiers in C.

5. Applications

The MIT Wearables Group has about thirty MIThril 2003 systems in active use, including applications within several collaborative class settings. Another fifty systems have been built and demonstrated for instrumenting large-scale classes for group-based applications.

The following are a number of real-world case examples of current, multi-user projects that are built upon various parts of the MIThril 2003 platform. The examples are chosen to demonstrate the modular, configurable nature of the MIThril 2003 hardware as well as the flexibility of the software architecture to accommodate a variety of high bandwidth, real-time applications.

5.1 Health Monitoring and MIThril Hardware

Using a variety of physiological sensors available, the MIThril 2003 platform lends itself naturally to be able to do a wide variety of clinical trial and long-term healthcare monitoring applications.

One such project involved a partnership with a Harvard/MIT neurologist to create a wearable system for classifying the medication state of Parkinson patients [28]. Using existing MIThril components, a working prototype apparatus was initially created in less than an hour. The Parkinson research project apparatus is an example of the flexibility of the MIThril 2003 hardware architecture. Its modularity and ease of reconfiguration makes prototyping applications easy, yet the resulting system is robust enough for real-world research applications. The ultimate goal of the study was to be able to develop a wearable real-time medication state analysis system, which is readily implementable using the MIThril 2003 framework.

As another example application, we are collaborating on the MIT/TIAX PlaceLab, a cross-institutional research living environment [17], to provide a platform to be able to collect and study long-term health information. Currently we have patients instrumented with Hoarder sensor boards to extract accelerometer information as well as IR tags instrumented in the house to profile location activity data over longer periods of time.

This information can then be used to construct activities of daily living, important information in being able to profile a person's healthy living style. Furthermore, these activities of daily living can initiate action on the part of the wearable PDA. Examples include experience sampling [13], a technique to gather information on daily activity by point of querying (which can be set to trigger based on movement or other sensed

context by the PDA). The system can also proactively suggest alternative healthy actions at the moment of decision, where it has been demonstrated as being more effective at eliciting healthy behavior.

5.2 OpinionMetrics and the Enchantment Whiteboard

OpinionMetrics is a set of applications designed to provide lecturers and teaching assistants with feedback about how students are tracking material presented in class. By selecting "Applause", "Bored", or "Lost" from the interface running on a WiFi equipped Sharp Zaurus, a student can affect the class' aggregate opinion, which the lecturer can view in real-time with alerts when thresholds are met. The lecturer can also poll students during the class to clarify the class' level of understanding on a particular topic.

The OpinionMetrics project uses the Enchantment Whiteboard system to manage the interactions of multiple, distributed applications. We are in the process of deploying our OpinionMetrics software into several classroom situations, including a Finance class at the MIT Sloan Business School with over forty students. Preliminary results show rich data sets from which the individual and aggregate state of the class can be derived, and participant surveys reflect positive user feedback. We anticipate that both the real-time feedback and the offline analysis of lectures will provide professors with useful information to meet the needs of their students.

5.3 Socio-Physiometric Feedback and the Enchantment Signal System

We are also deploying our systems in Sloan public speaking and negotiation classes this upcoming academic year, where it is useful to be able to monitor individual and group reactions to structured interactions. For the classes, the speaker and audience members will be outfitted with light-weight MIThril 2003 system that can measure skin temperature, EKG, and galvanic skin response. The MIThril systems are also configured to act as an accelerometer-based head nodding/shaking detection system, developed using the MIThril Context Engine framework described in Section 4.3). These systems will be used to stream physiologic/movement data in conjunction with the opinion data from the OpinionMetrics software.

The data will be used to gauge interest and agreement levels in real-time, and to be able to cross-compare self-reporting results to baseline information such as unconscious nodding in agreement and psychophysiologic cues such as heart rate and galvanic skin response, which is highly correlated with stress and sympathetic nervous system arousal.

The real-time visualization of the aggregated Enchantment signal information serves a dual-roll for the speakers and teacher. The audience's aggregated psychophysiology statistics can be used to gauge audience attention and interest in a form of socio-biofeedback. The speaker is also wired with his own physiological signals, which can be used as biofeedback for the speaker to identify his own mental state, or by the teacher, who can observe the effects of the speaking or negotiation interaction. Within the context of a structured interaction of a speaking or negotiation class, the person's performance can be studied in a very controlled environment, giving the individuals as well as the instructor valuable feedback.

5.4 Reality Mining and Conversation Analysis

Another interesting area of exploration is deploying the MIThril 2003 systems in group settings to capture conversations and develop statistics on the dynamics of group interaction. For this purpose, a student seminar class in Spring, 2003 called Digital Anthropology, organized and taught by Joost Bonsen, Nathan Eagle, and the authors, was instrumented with MIThril 2003 systems.

Every student was outfitted with a MIThril 2003 system, which recorded continuous high quality (16-bit, 22 khz) streaming audio using the Enchantment infrastructure to a remote server. Profiles of a participant's conversation such as speaking rate, energy, duration, participants, interruptions, transition probabilities, and time spend holding the floor were calculated using conversation detection and analysis algorithms from Nathan Eagle's Reality Mining framework. This information gives valuable insight to the context and content of conversation as well as capturing the dynamics of how such conversations are structured. One can even infer information such as the proximity of participants by comparing relative volume levels of a speaker's voice in the different audio streams [9].

6. Conclusions

Combined with the Enchantment/Signal and Context Engine software infrastructure, the MIThril 2003 architecture embodies a flexible system infrastructure capable of a variety of individual and group-based context-aware applications. As the various applications that have already been developed demonstrate, the MIThril hardware and software infrastructure allow for the rapid implementation of complex, distributed applications.

It is our hope that other people will find our infrastructure and software tools useful. As with all MIThril code, these tools are open-source and available under the terms of the GPL, along with our hardware

design files for the Hoarder sensor hub as well as other sensor designs [16].

7. Acknowledgements

We would like to recognize the following people and the rest of the Media Laboratory wearable community for their contributions to the MIThril 2003 platform and applications:

Vadim Gerasimov designed the Hoarder sensor hub. Ivan Chardin helped with the MIThril 2003 system integration and applications. Edward Keyes originated the Enchantment system concept. Nathan Eagle developed the audio processing algorithms and applications. Joost Bonsen originated the Digital Anthropology seminar as a forum to develop applications for the MIThril 2003 systems. Josh Weaver and David Klapper worked on the Parkinson Study.

8. References

[1] P. J. Brown, J. D. Bovey, and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace", *IEEE Personal Communications*, 4(5), October 1997, pp. 58—64.

[2] N. Carriero and D. Gelernter. "Linda in Context", Comm. ACM, 32(4), April 1989, pp. 444—458.

[3] T. Choudhury, A. Pentland, "The Sociometer: A Wearable Device for Understanding Human Networks", Technical Report, MIT Media Lab, http://web.media.mit.edu/~tanzeem/TR-554.pdf

[4] B. Clarkson and A. Pentland, "Unsupervised Clustering of Ambulatory Audio and Video" , *International Conference of Acoustics, Speech and Signal Processing*, Phoenix, Arizona, March 1999, pp. 3037 – 3040, vol 6.

[5] R. DeVaul, "Gaussian Mixture Model Tutorial", http://www.media.mit.edu/wearables/mithril/BNT/mixtureBNT.txt

[6] R. W. DeVaul, S. J. Schwartz, and A. S. Pentland, "MIThril: Context-Aware Computing for Daily Life", Whitepaper, MIT Media Lab, 2001.

[7] R. W. DeVaul and S. Pentland. "The MIThril Real-Time Context Engine and Activity Classification", Technical Report, MIT Media Lab, 2003.

[8] R. Duda, P. Hart, D. Stork "Pattern Classification", John Wiley & Sons Inc, 2001.

[9] N. Eagle, A. Pentland, "Social Network Computing", Submitted to: *The Fifth International Conference on Ubiquitous Computing* (UbiComp), 2003.

[10] V. Gerasimov, Hoarder Data Acquisition System, http://vadim.www.media.mit.edu/Hoarder/Hoarder.htm

[11] House_n MIT Home of the Future Consortium, http://architecture.mit.edu/house_n/

[12] IBM Almaden Research, "TSpaces", http://www.almaden.ibm.com/cs/TSpaces/

[13] S. S. Intille, C. Kukla, and X. Ma, "Eliciting User Preferences Using Image-Based Experience Sampling and Reflection", *Conference on Human Factors and Computing Systems*, April 2002, pp 738 – 739.

[14] M. Jordan, "Learning in Graphical Models", Kluwer Academic Publishers, 1998.

[15] V. Lesser, R. Fennell, L. Erman, and D. Reddy, "Organization of the Hearsay II Speech Understanding System" *Acoustics, Speech, and Signal Processing, IEEE Transactions*, Feb 1975 , pp 11—23.

[16] P. Lukowicz, U. Anliker, G. Troster, S. J. Schwartz, and R.W. DeVaul, "The WearARM Modular, Low Power Computing Core", *IEEE Micro*, May–June 2001, pp. 16—28.

[17] MIT/TIAX PlaceLab, http://architecture.mit.edu/house_n/web/placelab/PlaceLab.pdf

[18] MIT Wearables Lab, http://www.media.mit.edu/wearables

[19] K. Murphy, Bayes Net Toolkit, http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html

[20] J. Pascoe. "Adding Generic Contextual Capabilities to Wearable Computers", *Digest of Papers. 2nd International Symposium on Wearable Computers*, IEEE, October 1998, pp. 92–99.

[21] G. Picco, A Murphy, G. Roman, "LIME: Linda Meets Mobility", *Software Engineering*, May 1999, pp. 368—377.

[22] R. Rajkumar, M. Gagliardi, and Lui Sha, "The Real-time Publisher/Subscriber Inter-process Communication Model for Distributed Real-time Systems: Design and Implementation", *Real-Time Technology and Applications Symposium*, May 1995, pp. 66—75.

[23] Shafer, Stentz, and Thorpe, "An Architecture for Sensor Fusion in a Mobile Robot", *International Conference on Robotics and Automation*, IEEE, April 1986, pp. 2002—2011.

[24] T. Starner, B. Schiele, and A. Pentland, "Visual Context Awareness in Wearable Computing", In Digest of Papers. 2nd *International Symposium on Wearable Computers*, IEEE,, October 1998, pp. 50—57.

[25] T. Starner, B. Rhodes, J. Weaver, A. Pentland, "Everyday-use Wearable Computers", http://citeseer.nj.nec.com/319734.html

[26] Sun Microsystems, "JavaSpaces Technology", http://java.sun.com/products/javaspaces/

[27] M. Sung, VitaMon: Critical Health Monitoring, http://www.media.mit.edu/~msung/vitamon.html, 2002.

[28] J. Weaver, "A Wearable Health Monitor to Aid Parkinson Disease Treatment", MIT M.S. Thesis, June 2003.