Confidence Based Sensor Fusion for Resilience to Lossy Environments and Smart Power Savings

Scott Fisk

12 May 2011

The Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dan Siewiorek
Asim Smailagic

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

We describe an exploration of sensor fusion, power savings and personalization for Ergobuddy, a virtual coach system for package delivery workers to help prevent injury and reinforce ergonomic practices. The activities are inferred from a handheld device and supportive wearable devices connected in a wireless bluetooth network. A comparison of sensor fusion partitions is presented with measurements of bandwidth and accuracy. We utilize a dynamic confidence based fusion technique to counteract lost packets and node failures. 89% classification accuracy is achieved with resilient dynamic fusion method. Power savings as a result of intelligently using sensors is discussed. A comparison of lab data performance, real world scenario performance and real world scenario performance with personalization is performed. We conclude that training and testing solely on lab data is not indicative of a system's real world classification performance. A drop of 20% in accuracy is observed when utilizing lab generated models on data collected in a real world scenario. However, using ground truth we are able to simulate the personalization of the classifier to the user. The upper bound, assuming that all incorrect classifications are identified, is a 25% improvement over a classifier without personalization. We expect that this upper bound would be attainable in most real world systems given enough user-to-system feedback.

Acknowledgments

I want to thank my advisors Prof. Asim Smailagic and Prof. Dan Siewiorek for supervising my work, for their valuable inputs and for their help and support. I would also like to think Dan Sadler and Keshu Zhang for mentoring me through the Motorola and Carnegie Mellon relationship and providing me with an opportunity to intern at Motorola. Finally, I want to thank Brian French and Cem Onyuksel for many hours of help during the project and all the subjects for participating in the user study.

This material is based upon work supported by a grant from Motorola Incorporated, the National Science Foundation under Grant Nos. EEEC-540865 and 0205266, NIH grant on Real-Time Interview Technologies No.U01 023821, and Pennsylvania Infrastructure Technology Alliance (PITA), a collaboration among Commonwealth of Pennsylvania, Carnegie Mellon and Lehigh University.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Goals . . . 1
- 1.3 Outline . . . 2

2 Related Work **3**
- 2.1 Activity Recognition . . . 3
  - 2.1.1 Power Efficiency . . . 3
  - 2.1.2 Sensor Fusion . . . 4
- 2.2 Workplace Deployment . . . 4
- 2.3 Personalization . . . 4

3 Preliminary Exploration **6**
- 3.1 A First Look at Data . . . 6
- 3.2 Context Recognition Pipeline . . . 7

4 Software Design **9**
- 4.1 Architecture . . . 9
  - 4.1.1 Partitions . . . 10
- 4.2 Demonstration Platform . . . 11
  - 4.2.1 Retraining . . . 12
  - 4.2.2 Dynamic Sensor Set . . . 12
  - 4.2.3 Realtime Visual and Auditory Feedback . . . 12
  - 4.2.4 End of Day Analytics . . . 13

5 Proposed Methods **14**
- 5.1 Hardware System . . . 14
- 5.2 Data Set . . . 14
- 5.3 Processing . . . 15
  - 5.3.1 Data Windowing and Labeling . . . 15
  - 5.3.2 Feature Selection . . . 15
- 5.4 Algorithms . . . 15
  - 5.4.1 Machine Learning Parameters . . . 16
    - 5.4.1.1 K nearest neighbors . . . 16

5.4.1.2 Decision Trees 17
5.4.1.3 Perceptrons 17
5.4.1.4 Decision Trees using Adaboosting 18
5.4.1.5 Random Forests 18
5.4.1.6 Support Vector Machines 18
5.4.2 Sensor Fusion 18
5.5 Testing 19
5.5.1 Tolerating Faults 19
5.5.2 Saving Power 20
5.5.3 Personalization 21
5.5.3.1 Data Fidelity 21
5.5.3.2 Personalization Technique 22

6 Results and Discussion **24**
6.1 Experimental Issues 24
6.2 Activity Space Coverage 24
6.3 Classification 25
6.3.1 Accuracy 26
6.3.2 Memory Footprint 27
6.3.3 Classification Delay 27
6.4 Fusion Methodology 28
6.5 Resilience 28
6.6 Energy Savings 29
6.7 Tradeoffs 31
6.8 Customer Scenario 32
6.9 Personalization 33
6.9.1 Scenario without Personalization 33
6.9.2 Scenario with Personalization 34

7 Conclusion **36**
7.1 Contributions 36
7.2 Future Work 37
7.2.1 Implementation Issues 37
7.2.2 Further Research Questions 37

References **39**

Section 1

Introduction

Context aware computing is the use of sensors and learning algorithms to identify the user's state and modify the system's behavior based on this information. It is often applied to wearable computing since the system is physically attached to the user.

1.1 Motivation

Activity recogntion is a popular application of context aware computing with many applications. In this work activity recogition will be applied to the realm of package delivery and warehouse work. Package delivery offers unique advantage as a realm of application because it has both a strong need for such a system and a population of workers that allows for easy deployment. The package delivery occupation is an injury prone field. Transportation and warehousing workers experience the highest injury and illness rate out of any service-providing industry sector. The U.S. Bureau of Labor Statistics reported that in 2006 6.5% of 3.9 million full time workers in this field experienced injuries or illnesses [1]. These work related injuries are expensive costing about $1360/year per worker and 2.9% of total compensation [2]. The ability to reduce injuries, and log the context in which injuries occur is valuable information for establishing interventions with economic impact. Package delivery workers are required to wear company mandated uniforms, reducing compliance issues in deploying multi-sensor systems.

1.2 Goals

The primary goal of the described work is to analyze the feasibility of using machine learning and body worn sensors for activity recogntion in the package delivery occupation. In order to achieve this goal a unique dynamic context recogntion engine was developed that allows for increased power efficiency and increased environmental resiliency. The performance of various machine learning algorithms is considered in order to determine the optimal algorithm for the embedded environment that Ergobuddy would be deployed. The accuracy, memory footprint, and decision latency are the primary performance metrics when evaluating algorithms alone.

Additionally we explore personalization as an effective technique to maximize classifier performance for activity recognition. A comparison of lab experiments, scenario experiments and scenario experiments with personalization are performed to quanity this performance. A measure of accuracy improvement as a function of retraining frequency is an interesting relationship we hope to illustrate.

1.3 Outline

This report is split into seven sections.

Section 2 describes the related work to the experiment descibed in this report. It covers the areas of activity recognition, sensor fusion and workforce deployment. Section 3 describes the initial exploration into this problem. Section 4 describes the software steps involved in the classification process. It also describes the techniques for generating the optimal machine learning parameters and the compared techniques to fuse local decisions together. Section 5 describes the setup of the experiment, how the approach addresses the primary concerns of the study, and a description of the hardware. Section 6 describes the results of the multi user study including activity space coverage, energy consumption, resiliency, and per sensor model performance (accuracy, decision speed, memory footprint). It also decribes the results of the personalization experiments. Section 7 describes the contributions of this report and future work opportunities.

Section 2

Related Work

The proposed system, called ErgoBuddy, utilizes multiple sensors in a power efficient and dynamic way. The field of activity recognition is rich with work using accelerometers. The accelerometer activity recognition work will be discussed, followed by an overview of sensor fusion and projects that deploy activity recognition in the workplace.

2.1 Activity Recognition

A recent review of activity recognition that covers the extensive depth of the field is Bao and Itelle[3]. An important aspect of this review is that it identified the common flaws in activity recognition studies and the limitations of deploying a system developed in the lab into a wild environment. The primary issues identified were: too few subjects in the user studies, an overly constrained set of activities to be identified, and a lack of real-world conditions in lab studies due to unforeseen influences. The authors of[3]attempt to address these issues in their own study and achieve 80% recognition accuracy. This accuracy is lower than most work in the activity recognition field, but is due to the more realistic conditions of their study.

2.1.1 Power Efficiency

Various power efficient multi-sensor systems and algorithmic techniques exist for activity recognition. In [4] a methodology is described to characterize a systems functionality, power usage and packaging. Specifically, this work introduces metrics that illustrates a design's performance against these three characteristics. Designers can then choose which of the aspects they would like to emphasize.

Power minimization approaches include strategies to reduce sampling rate [5] and other techniques [6], including sensor selection, feature selection, frame length (number of samples per window), and sliding window step size. These parameters have a direct effect on power consumption by reducing computational complexity. Neither of these papers consider the selection of a subset of available active sensors, but [7] has investigated this and offered a solution that allows the user to extend system uptime while guaranteeing a minimum level of accuracy. Ergobuddy explores similar techniques, but with a virtual coach as an end goal. Other works are primarily concerned with immediate feedback, but Ergobuddy focuses on obtaining information useful for end-of-day analytics that can discover bad habits. These analytics can function with lower recognition accuracy, but uptime and user compliance become more important.

2.1.2 Sensor Fusion

Sensor fusion is a popular technique in robotics [8], military applications [9] and activity recognition [10]. In the activity recognition realm Laerhoven et al. [10] studied the potential effectiveness of fusion by measuring mean dispersion of accelerometer data based on the number of sensors and the number of activities. The study used a distributed fusion approach stating that it provided superior scalability, flexibility and robustness. Follow up studies, especially those performed by Zappi et al. [7], further developed the concept of using a distributed fusion scheme. Zappi et al. applied a classifier to an automobile inspection dataset consisting of 10 gestures repeated 19 times by a single user. Gestures included writing on a notepad, opening hood, check steering wheel, etc. Important contributions include measures of scalability and robustness, which are evaluated by accuracy as a function of the number of sensors and the number of noisy sensors respectively. Three types of faults were considered: sensor loss, rotational noise (placement errors), and random noise (random responses). Fusion is performed by using majority voting and a naive Bayesian technique.

There are two approaches to classifier fusion, methods that support dynamic sensor sets, i.e. sets that can have sensors added and removed during runtime, and those that require a static set. Majority voting is a simple example of a dynamic fusion technique and naive bayes is a static fusion technique. We will utilize what we call confidence based fusion, which is a dynamic technique based on a confidence distribution generated locally on every sensor. This allows the master device performing the fusion to have no prior knowledge about the sensors it is communicating with.

2.2 Workplace Deployment

Another important and relevant area of related work is applying activity recognition in the workplace. Much work has been done by Lukowicz et al., most recently, [11], as part of the wearIT@work project, a long term wearable computing project with several industry partners. The wearIT@work projects has developed and deployed systems in Healthcare, Emergency Rescue, Aircraft Maintenance and Production Management and Training. They are primarily concerned with wearable computing, specifically measuring the effectiveness and applicability of the wearable systems.

2.3 Personalization

Work has been done on the personalization of classifiers for physiological data[12]. Biological signals, being innately more individualized require personalization, but certain applications of activity recognition can also benefit from such a technique.

Activity recognition usually tries to generalize classification models. Virtual coaches are different in that they are used by the same user day in and day out. The user has plenty of opportunities to provide feedback to the system to ensure coaching is done effectively. Additionally, virtual coaches, especially Ergobuddy, focus on obtaining information useful for real time feedback and high accuracy through personalization ensures such a system will not become untrusted or ignored.

Section 3

Preliminary Exploration

3.1 A First Look at Data

To understand qualitatively how accelerometer responded to different body locations a short experiment was performed. Since one of the applications of Ergobuddy is to identify incorrect or unsafe lifting procedures we had a single subject lift a box with his back (incorrect) and with his legs (correct). The magnitude of the acceleration vector was plotted over time. Figure 3.1shows the results of this experiment. The handheld accelerometer, which was set down on the box during the lifting procedure, provided no discernable difference when the box was lifted correctly vs incorrectly. However, the hip accelerometer provides a clear difference between the two lifts. A critical finding here was that multiple sensors would definitely be needed and that the performance of the position will vary greatly depending on the activity being performed.

As expected the acceleration magnitude was approximately 9.8 $m/s^2$when no activity was being performed, but the calibration was different per sensor (the Hip and Handheld accelerometers were different physical devices). Thus normalization of the data is required.

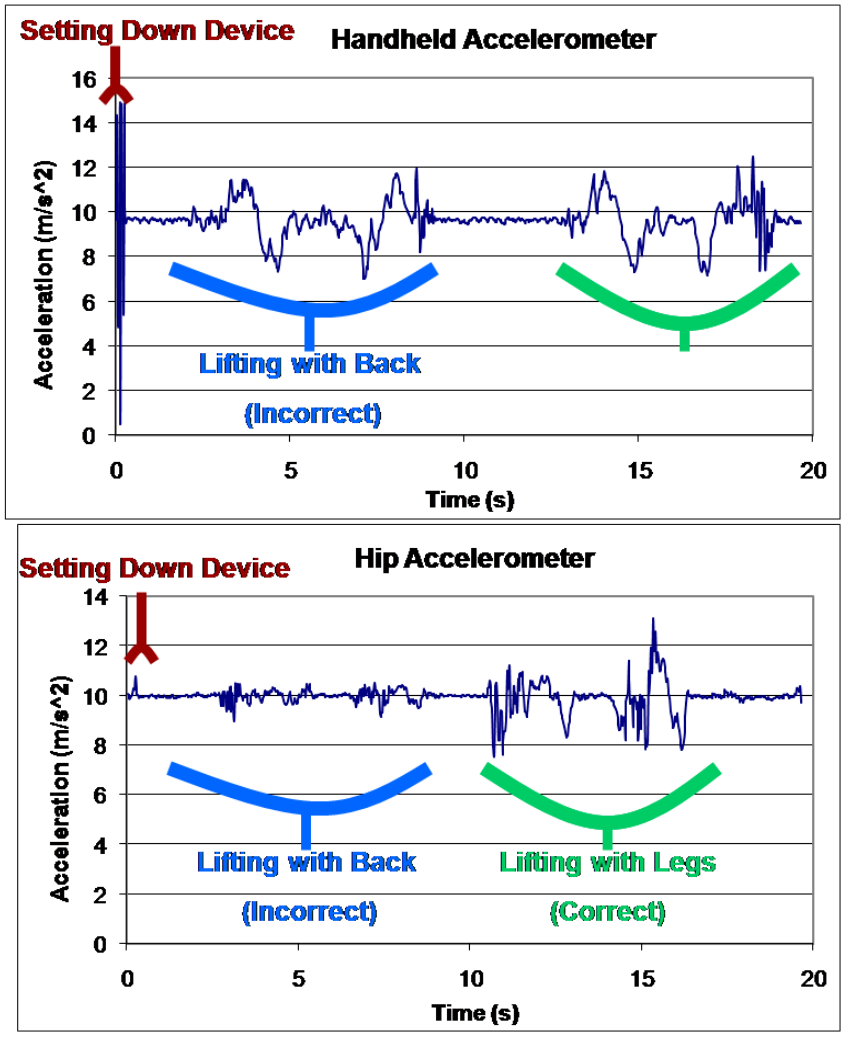

Figure 3.1: A comparison of two locations to evaluate the "gesture" produced when a box is lifted both correctly and incorrectly.

3.2 Context Recognition Pipeline

Based on previous activity recognition research and an analysis of our own problem a simple four step context recognition pipeline was developed as shown in Figure 3.2. This pipeline consists of four stages described from bottom to top:

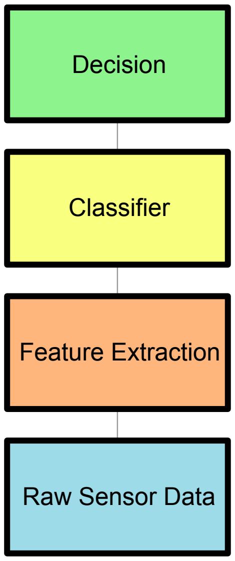

Figure 3.2: 4 Step Context Recognition Pipeline

Raw Sensor Data This is simply the raw data stream. In our experiments it is 3 axes accelerometer data. Since this pipeline is meant to be generalized across any sort of context recognition device this layer could be any sort of sensor data, generated from hardware or software.

Feature Extraction The feature extraction step is where raw data is transformed into usable features that can be input to a machine learning model. From our findings above, it was clear that normalization of this data should be performed. From previous work [13] statistical features such as mean, variance, etc would be quick and effective calculations to perform on the raw data during this step.

Classifier The classifier step is where features are transformed into a labeled context. This requires a training process in which data is collected and labeled. Labeled data that has already undergone the feature extraction process is passed to machine learning algorithms which discriminate between labeled contexts based on the features. There are many machine learning algorithms available and also many tools simplifying their use. Weka [14] and libSVM[15] are tools utilized to some extent during the Ergobuddy project.

Decision The decision can be obtained a number of ways. Information from the classifier can be used to make a more intelligent decision, or information from the classifier can be taken at face value and simply passed through this layer. We have introduced sensor fusion in the related work section which is a more intelligent use of classifier information and thus a sensor fuser would reside in this layer.

Section 4

Software Design

4.1 Architecture

The modular architecture described in Section 3 is broken into four steps that can be deployed on multiple devices based on application needs. This is critical to the dynamic nature of the system and encourages the use of sensor fusion and per device power utilization techniques. There are no contraints on where each of these layers must reside, but the selected configuration will have a large impact on the performance of the system. Figure 4.1 shows three of the possible paritions of the four layers each with different reprecussions on the system performance. The specifics reprecussions will be discussed in the next section. For example, if one chooses to deploy the raw sensor data layer on the sensor device and the rest of the layers on a personal computer a heavy bandwidth burden would be put on the link between the sensor device and personal computer. This is illustrated as the Centralized Architecture in Figure 4.2.

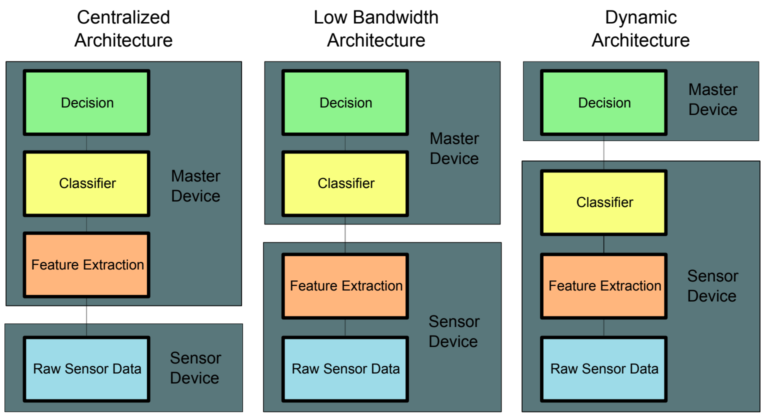

Figure 4.1: Partitions of Context Recognition Pipeline

4.1.1 Partitions

Centralized Architecture

An illustration of this architecture is available in Figure 4.2. This is a commonly used architecture in which raw data from all nodes in a network are transmitted to a master device for feature extraction and classification. In our experiment the amount of data transmitted is a continuous stream at 2 KB/sec. In addition to the higher bandwidth requirement this scheme requires a static set of sensors. Since all raw data is directly available to the classifier, in a perfect environment we expect the highest accuracy from this scheme. However, the loss of a single sensor can dramatically reduce accuracy since the classifier is trained on data from a complete sensor set.

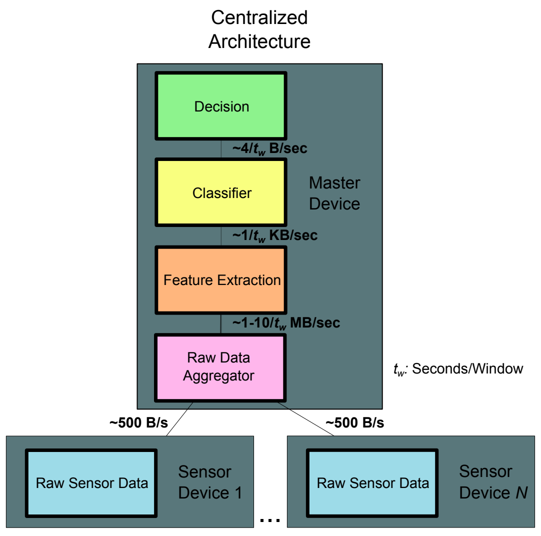

Figure 4.2: Centralized Architecture

Low Bandwidth Architecture

An illustration of this architecture is available in Figure 4.3. This architecture requires lower bandwidth then the centralized architecture since feature vectors are transmitted upon completion, yielding approximately 200/t_w Bytes/sec, where t_w is the window length of a classification. This yields the same accuracy as a centralized aggregation architecture unless the feature extraction technique is changed. Again a static set of sensors is required.

Dynamic Architecture

An illustration of this architecture is available in Figure 4.4. The dynamic architecture further decreases bandwidth since only the context and confidence information is transmitted every window. This is approximately 80/t_w Bytes/sec. Decision fusion differs from the previously described aggregators in that it allows the number of sensors to be dynamic. Accuracy is expected

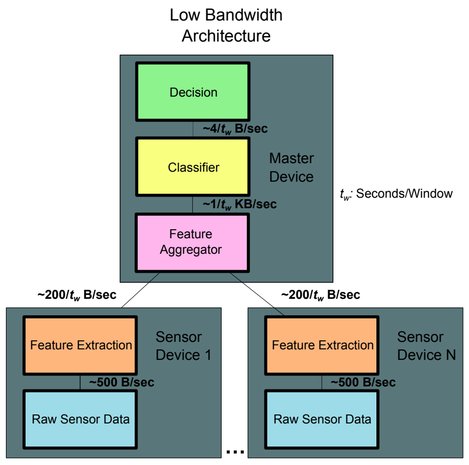

Figure 4.3: Low Bandwidth Architecture

to decrease because only an abstraction of the raw data is provided for the final decision maker (fuser), but the system is resilient to sensor failure and packet loss faults since the final decision is only based on classifications from the available sensors.

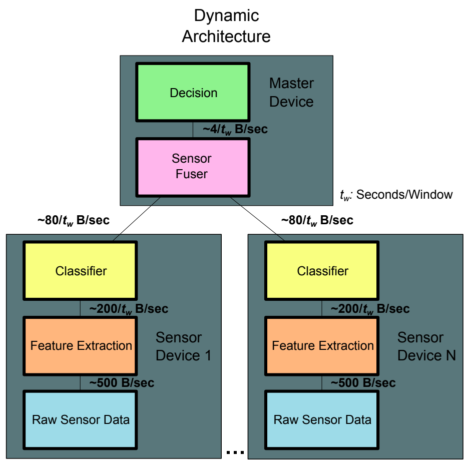

Figure 4.4: Dynamic Architecture

4.2 Demonstration Platform

Ultimately the Ergobuddy system will be demonstrable and deployable. Currently this is an ongoing process. A demo of this system supports the use of our dynamic activity recognition engine by non technical users. The demonstration systems will include four major features that cover initial customer use cases for package delivery or warehouse work.

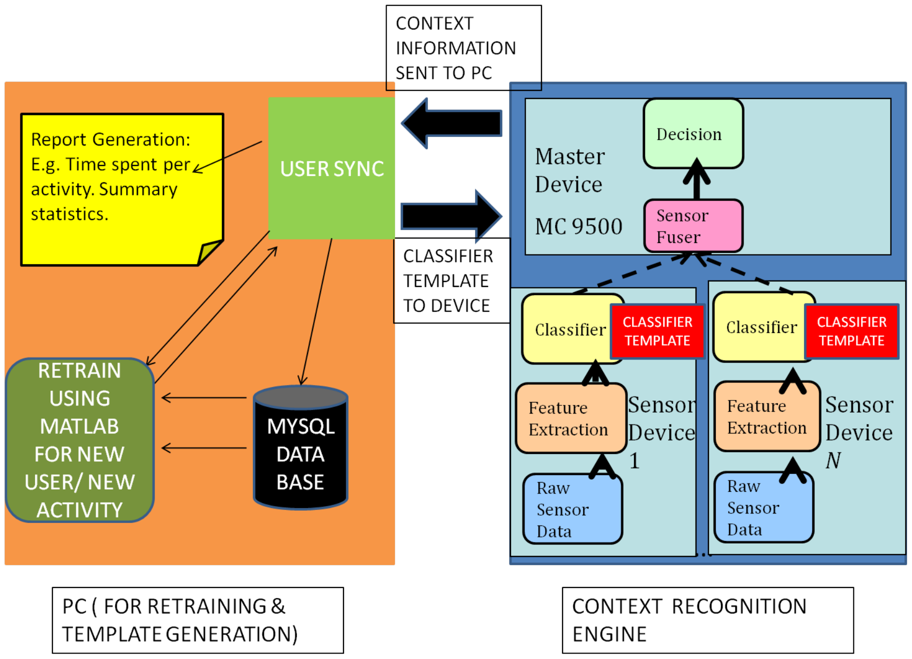
CONTEXT INFORMATION SENT TO PC
Report Generation: E.g. Time spent per activity. Summary statistics.
USER SYNC
RETRAIN USING MATLAB FOR NEW USER/ NEW ACTIVITY
MYSQL DATA BASE
CLASSIFIER TEMPLATE TO DEVICE
Master Device MC 9500
Decision
Sensor Fuser
Classifier
CLASSIFIER TEMPLATE
Feature Extraction
Sensor Device 1
Raw Sensor Data
Classifier
CLASSIFIER TEMPLATE
Feature Extraction
Sensor Device N
Raw Sensor Data
PC (FOR RETRAINING & TEMPLATE GENERATION)
CONTEXT RECOGNITION ENGINE

Figure 4.5: Master Device and PC Interaction for Retraining process

4.2.1 Retraining

The software allows the user to collect data, add that data to a new or existing activity dataset, store the data in a database, and then retrain the models based on the self annotated data. The complete process, which is an interaction between the PC and the master device is shown in Figure 4.5. Information collected on the device during a retrianing mode is sent to the PC, then the data to be included in the retrained models is selected and the models are built. Then the models are transmitted back to the master device which distributes the models to the corresponding connected devices.

4.2.2 Dynamic Sensor Set

Consistent with the vision that sensors can be added and removed from the system at will, we will utilize a dynamic architecture in the deployed system. As described this will allow the addition or removal of sensors willingly or unwillingly in the case of sensor failure.

4.2.3 Realtime Visual and Auditory Feedback

On the handheld device information will be provided to the immediate user in two ways. The user can access visual data, which includes a break down of activities performed up to that point in the work day. This would be data similar to what will be transmitted back to the PC at the end of the work day. He or she will also be able to enable auditory feedback. Conditions can be programmed by the user to enable this feedback. An useful example of auditory feedback would be the notification of the user when a certain amount of time has passed while lifting. Overuse and repetitive use injuries could be reduced using this feedback.

4.2.4 End of Day Analytics

The system will be collecting relevant contexts during the day and at the end of the day these statistics can be transmitted back to the PC and stored. Managers and employers would be interested in this information to streamline employee workdays and again reduce possible injuries by breaking up dangerous periods of work. Long term statistics can also be collected to evaluate job performance.

Section 5

Proposed Methods

5.1 Hardware System

The experimental setup uses a combination of seven devices to recognize user activities. Six wearable wireless sensors communicate with a handheld master device, which logs and outputs the user's activity.

The master device is a Motorola MC9500 Industrial-Class Rugged Mobile computer, targeted for use by enterprise employees such as package delivery workers [16]. It has a 3D accelerometer onboard and communicates with slave devices over bluetooth. A second MC9500 acts as a slave and sits in a holster on the user's hip. A delivery worker would normally only carry one MC9500, either in his hand or in a holster, but a second device allowed data from both positions to be collected simultaneously.

The other five devices are eWatches. eWatches are low-power wearable sensor platforms with limited computational power. They have onboard 3D accelerometers and communicate with the master device over bluetooth [17]. These sensors are worn on the wrist, ankle, upper arm, lower back, and as a lanyard around the neck.

5.2 Data Set

The system was trained on 12 users with a set of 10 activities: sitting, standing, walking, running, lifting, carrying, sweeping/mopping, using stairs, using a ladder, and using a cart. The users performed each activity for a period of two to four minutes, depending on the activity, and all sensor data was annotated and synchronized. Approximately 20 hours of data was collected in total. Data is synchronized based on bluetooth annotations and normalized on a per sensor basis to account for different accelerometer calibrations.

5.3 Processing

5.3.1 Data Windowing and Labeling

The data collected is a time series dataset, thus segmentation needs to be performed to allow for classifiable windows. Although Hidden Markov Models(HMM) are usually ideal for a time series experiment there was not sufficient natural data in the dataset to model the probabilities of state transitions. For this reason we experiment with various windows and calculated the features described in Section 5.3.2 on each window. Labeling for each window were derived from the labels of the raw data making up that segment. Windows were not intermixed with multiple labeled activities.

Window length cannot be selected solely to provide the best performance since it is directly correlated with decision latency, the delay in the recognition of an activity after its performance. Latency in this case is at max the length of the window, but could be much lower. Window length also depends on the length of the dataset and length of time activities are trained. Since some of the activities trained for the classifier were only trained in 30 second blocks, we limited our window length to be either four or six seconds.

5.3.2 Feature Selection

The features described in Table 5.1 are calculated on each window. These statistical features are common in literature.

Feature	Description	Sensor Used
MEAN{X,Y,Z,M}	Mean	Accelerometer
VAR{X,Y,Z,M}	Variance	Accelerometer
MAD{X,Y,Z,M}	Mean Absolute Deviation	Accelerometer
ZCR{X,Y,Z,M}	Zero Crossings	Accelerometer
MCR{X,Y,Z,M}	Mean Crossings	Accelerometer
RMS{X,Y,Z,M}	Root Mean Square	Accelerometer

Table 5.1: Features Utilized

5.4 Algorithms

For completeness and because the classification process can take place on devices with different computation abilities six different algorithms were compared. They include:

- K-nearest neighbors (kNNs) [18]
- Decision Trees [19]
- Perceptron [20]
- Decision Trees with Boosting [21]
- Random Forests [22]

Algorithm 5.1 Parameter Optimization Procedure

```
for all $sensors$ do
    for all $algorithms$ do
        $result \leftarrow LOSOCV(dataset)$
        $algorithms.sensor.params \leftarrow results.best\_params$
    end for
end for
```

- Support Vector Machines[23]

The performance of these algorithms depend on window size, a globally optimized parameter, that implies performance characteristics independent of algorithm, and two locally optimized parameters, which differed depending on the algorithm being optimized. The locally optimized parameters are described in the next section.

5.4.1 Machine Learning Parameters

In order to train the optimal model for each subject the procedure described in Algorithm 5.1 was used.

LOSOCV, or leave one subject out cross validation, was used to optimize the parameters outlined in Table 5.2 and described below.

Algorithm	Parameter 1	Parameter 2
kNN	Neighbors	Training Set Examples
Decision Trees	Pruning Factor	Minimum Instances per Leaf
Perceptrons	Weights	Momentum
Adaboosting	Pruning Factor	Minimum Instances per Leaf
Random Forests	Number of Trees	Number of Features
Support Vector Machines (SVM)	Cost	Gamma

Table 5.2: Optimized Parameters

5.4.1.1 K nearest neighbors

K nearest neighbors simply places training instances in a D dimensional space where D is the number of features. Then, by specifying the number of nearest neighbors to look at, K, one may make a classification decision. The two paramters optimized were the number of neighbors and the maximum number of instances allowed in training pool. Since the distance has to be calculated to all points anyway, the number of neighbors chosen will not have a significant impact on performance speed. Conversely the number of training instances to include in the model has a significant impact on both the size of the model and the speed of classification. This is intuitive since more training points requires more storage and more distance calculations per window.

5.4.1.2 Decision Trees

Decision trees are generated by finding feature splits that offer the greatest increase in Information Gain. Information Gain is defined as the change in entropy from one state to the next[24]. Since a decision tree can make infinite splits for continuous features we employ pruning to prevent overfitting. The two parameters optimized for the decision tree model were confidence factor and the minimum number of instances per leaf. Confidence factor is the parameter that controls pruning and a smaller value yields more pruning. The minimum instances per leaf simply limits how small a final node in the tree can be, again to prevent overfitting. The size of trees can vary greatly depending on how much pruning is performed. The size of the tree determines the size footprint the model, as more nodes and leaves require more memory to store. The size of the tree, specifically the depth, determines and the speed of classification as more branch statements will need to be performed.

5.4.1.3 Perceptrons

Perceptrons are a form of neural network in which the input vectors are weighted to perform optimal classifications. The two parameters optimized for perceptrons were the increment of the weights per iteration and the momentum applied during weight updating. These parameters vary the speed of which the algorithm converges to a solution, but do not have significant impact on the memory or computation requirement of the completed model. We utilize multilayer perceptrons so an additional weighting layer is added, a visual representation of multilayer perceptrons is provided in Figure 5.1. Unfortunately designing an optimal perceptron is more difficult than the other algorithms considered in this study. Results will show that performance is slow and accuracy is low when the necessary expertise is not available. Optimizing neural networks further was deemed beyond the scope of this study.

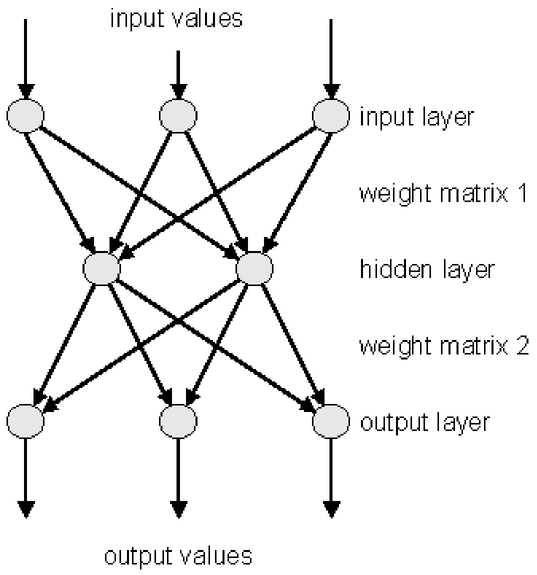

Figure 5.1: Multilayer Perceptron [25]

5.4.1.4 Decision Trees using Adaboosting

Boosting optimizes the same parameters as decision trees, but iteratively refines the tree based on incorrectly classified instances. The number of trees, or weak classifiers, in our experiments is held constant at a value providing reasonable model generation times.

5.4.1.5 Random Forests

In our application random forests refers to an ensemble of decision trees. A random subset of features is used to generate each tree and then multiple trees are used to vote on a final decision. Once the features are selected each individual tree is generated as described in Section 5.4.1.2. The two parameters optimized for random forest models are the number of trees to be generated and then number of features used in random feature selection. As expected the number of trees will negatively affect both the size of the model and the speed of classification. The number of random features selected also increases the complexity of each tree and thus increases the size of models and decreases speed of classifications. Random forests generate the largest models out of any of the algorithms considered in this study.

5.4.1.6 Support Vector Machines

Support vector map the data in D dimensional space where D is the number of features. A kernel is often applied to transform the data into a more discriminatory space. For our experiments we use an RBF kernel because [26]suggests that it can handle non-linearities better than a linear kernel and, unlike sigmoid or a polynomial kernel, it is impossible for the kernel to map input data to infinity, zero or invalid values. The optimized parameters are the cost factor and the kernel parameter, gamma. The time required for convergence of an SVM algorithm may be much higher that other considered models, but SVMs are able to reduce the input samples into a limited set of support vectors and the model size can be quite small.

5.4.2 Sensor Fusion

In order to leverage our dynamic architecture the fusion technique must be robust to dynamic sensor sets. The naive bayes technique used in [7] requires a static sensor set, and is not considered here. Polikar at al. [27] describes a number of dynamic fusion techniques and we consider five of them including the popular straight vote, max probability and rank [28] techniques. The sensor generated confidence data is fused using average and product techniques.

Confidence data is static and determined during the training of the classifier. Since all implemented algorithms are based on decision trees the confidence can be calculated at the terminating leaf as:

$$w^{\tau}(i) = \frac{\#instancesofi}{\#instances}$$

Where $\#instancesofi$ is the number of correctly classified training instances for activity i at the terminating leaf and $\#instances$ is the number of training instances at that leaf.

Since random forest and boosting algorithms run the training instance through multiple trees the final confidence vector is a summation over set T for each tree t:

$$w^{\tau}(i) = \frac{\sum_{t \in T} \#instancesofi}{\sum_{t \in T} \#instances}$$

The combination of $w^{\tau}$is performed on the master device using the following techniques. The activity with the highest score is the one ultimately chosen. R is the set of N sensors used and w^{τ} is the vector of confidence probability for each sensor τ. For an activity i, the fusion score $s(i)$ is described as:

- **Average**

$$s(i) = \frac{1}{N} \sum_{\tau \in R} w^{\tau}(i) \tag{5.1}$$

- **Product**

$$s(i) = \prod_{\tau \in R} w^{\tau}(i) \tag{5.2}$$

- **Straight Vote**

$$s(i) = \sum_{\tau \in R} f(i), f(i) = \begin{cases} 1 & : \arg\max_j w^{\tau}(j) == i \\ 0 & : \text{otherwise} \end{cases} \tag{5.3}$$

- **Max Probability**

$$s = \max_i \max_{\tau} w^{\tau}(i) \tag{5.4}$$

- **Rank** The final score is calculated as an average (Eq 5.1) after the following transform is performed on the sorted confidence vector τ.

$$w^{\tau}(i) = 1 - \frac{\tau(i) - 1}{|\tau|} \tag{5.5}$$

5.5 Testing

As described in the introduction, the primary goals of this experiment is to utilize multiple sensors and provide the highest possible accuracy while remaining resilient to sensor faults and saving power when possible. The aforementioned sections have outlined the process of optimizing accuracy, and below we describe the considerations for resiliency and power savings.

5.5.1 Tolerating Faults

Since the Ergobuddy system is to be deployed in harsh environments and expected to operate we must first consider the possible faults that can occur.

- Sensor node failure [7, 10]

    - This includes any situation in which a node could fail, both permanently or intermittently.

- "Stuck-at" classification [29]
    - This includes any situation in which a node is repeating decisions. This could be caused by sensor malfunction, providing invalid raw data to the classification engine.

- Transient Faults
    - Wireless packet loss
        - This situation occurs when a wireless transmit fails, could be due to range issues, interference or collision.
    - Sensor random response [7]
        - This situation occurs when a sensor is malfunctioning or memory utilized by the classification engine is malfunctioning, distorting the data or features.

Tolerance of these faults have been considered in previous literature as denoted. Faults that specifically afflicted our implementation (sensor node failure and wireless packet loss) are considered in this study.

5.5.2 Saving Power

For Ergobuddy to be useful, it must be able to run for long periods of time, providing reasonable recognition accuracy. To save power, the system can dynamically disable eWatch sensors at the cost of accuracy. Power usage of the master device was not considered because it must stay active for it's primary task of supporting the package deliverer, thus the power vs accuracy measurements only account for the five wearable eWatch sensors. eWatches can be put into a deep sleep state and woken up by a local hardware timer. When actively performing context recognition, each eWatch uses 149mW. When in deep sleep each eWatch uses 14mW. Power reduction is achieved dynamically by disabling incorrect sensors for a period of time, called penalty time. If a sensor's vote is not in line with the final decision computed by weighted voting, then a penalty time is enforced. Analysis includes an experimental sweep of penalty time from 0 to 100 seconds. Diminishing returns are described as the penalty time for which only 10% of the initial power savings or accuracy loss begins to occur.

Since the experiment considers temporal data a more realistic dataset was collected for additional power and accuracy experiments. This dataset consisted of four users who were instructed to perform a package delivery scenario (Table 5.3). The scenario is designed to simulate the start of a typical campus package delivery day, with the user moving seamlessly between activities.

Activity	Time Performed (s)	Label
Sitting w/ morning coffee	120	Sitting
Walking to stairs	60	Walking
Descend stairs	40	Stairs
Walking to truck	15	Walking
Load boxes into truck	120	Lifting
Dead time	-	Dead Time
Unloading boxes onto cart	120	Lifting
Move cart to 1st delivery	240	Cart
Carry 1 box to office	60	Carrying
Return to cart	60	Walking
Stand and talk to friend	120	Standing
Move cart to 2nd delivery	120	Cart
Deliver 4 boxes	30	Lifting
Walk 1 box to office	60	Carrying
Ascend stairs	40	Stairs
Check device	60	Standing

Table 5.3: Package Delivery Scenario

A simulator was designed to analyze the efficiency of a power saving algorithm on scenario data. The simulator uses the data collected in the lab to train models for each sensor location as described above. Scenario data is replayed by software, allowing the power saving algorithm to dynamically schedule downtime of sensors. Since the scenario data is labeled the performance of the power saving algorithm can be measured.

5.5.3 Personalization

5.5.3.1 Data Fidelity

The goal of personalization is to identify an upper bound of performance when a retraining technique is applied to more realistic scenario data. There are three types of data fidelity to consider when calculating the performance of activity recognition models: Lab Data, Scenario Data, and Wild Data. These three types are described below. The fidelity of this data varies as a function of the realism. Typically the more realistic the data is, the worse the fidelity of the data and the worse an activity recognition engine will perform.

Lab Data A typical starting point in building an activity recognition engine is to collect a fixed amount of data for each activity performed by each subject. This data does not need to be in any particular order, but the amount of data for each activity should be consistent to prevent bias in the trained classifier.

Performance is typically measured using “leave one subject out" cross validation accuracy in which N folds are performed, where N is the number of subjects. Each model is trained on N-1 subjects excluding the testing subject. Classification is then performed on samples from the testing subject with a resulting accuracy. Accuracy across all N folds is averaged to get an approximation of best case performance of the classifier. Since the data collected in the lab is clean and scripted, accuracy is quite high.

Scenario Data Scenario data is an attempt to emulate wild data. Subjects are asked to perform activities seamlessly in a natural order. An onlooker annotates when activity transitions occur so that ground truth labels can be generated.

Evaluation is straightforward, models are generated using the Lab Data and classification is performed on Scenario Data. Accuracy will usually diminish significantly when seamless and natural data is tested.

Wild Data Wild data is what would be collected if the system was attached to a user and he or she was asked to self annotate activities. There would be a significant amount of noise introduced in the ground truth labels since users often forget to annotate or annotate incorrectly.

Evaluation can be performed by training on both scenario and lab data and testing on wild data. Since the label noise compounds the error in the classifier the performance of such a system is further diminished from the scenario case.

5.5.3.2 Personalization Technique

In this report we examine the performance lost when moving from a lab data set to a scenario data set. We offer techniques to regain lost performance when making the transition to the more realistic scenario dataset. This technique is specifically to offer better performance for the immediate user of the system as it personalizes based on the performance of the activities by that single user.

Since ground truth is available in the scenario dataset it is easy to determine when an incorrect classification occurs. A retraining process occurs after a buffer of incorrect classifications is filled. Samples of the same label are randomly selected and removed with a 1:1 ratio in order to prevent the model from getting too large. After the retraining process completes the classification engine continues with increased performance. An illustration of this process is available in Figure 5.2.

The classifier that best suits this personalization scheme is kNN. Since kNN classifiers require no computation to retrain our technique is virtually free. The only cost is the computation time it takes to remove a sample from the dataset and add a new one. For this reason the buffer size offering the best accuracy increase should be chosen. When using parametric algorithms, or those that require computationally intense training to generate a model one would need to be more selective with buffer size.

Personalization is performed using the same scenario as the power saving experiment. A separate simulator was designed to analyze the efficiency of the personalization algorithm.

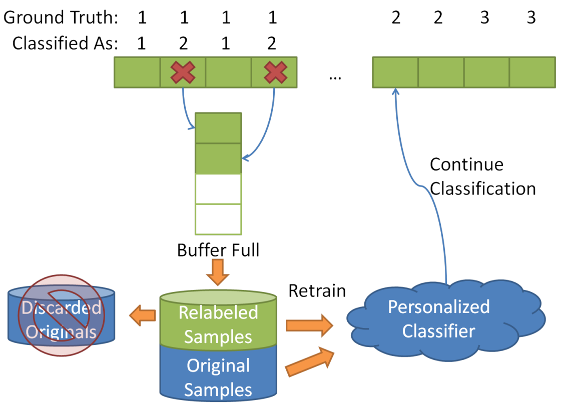

Figure 5.2: Retraining Process which triggers upon a full buffer of incorrect classifications

Section 6

Results and Discussion

6.1 Experimental Issues

During preliminary data collection it was determined that streaming accelerometer data to a master device from six sensors simultaneously at 20 Hz was unreliable. The antenna interference combined with the low power radios of the eWatch were likely sources of error. Since streaming was not feasible, the low bandwidth (Figure 4.3) and dynamic (Figure 4.4) architectures were evaluated. A low bandwidth architecture would have addressed the streaming bottleneck, but data collection indicated that nearly 15% of activity annotations were missed, correlating to 15% packet loss.

A static classifier, such as the one proposed for the low bandwidth architecture, would be unable to make a decision when afflicted by missing packets, since a significant portion of input features to the machine learning algorithm are missing. In the low bandwidth architecture each supporting sensor has an 85% reliability and the master sensor is assumed to be 100% reliable as it is required for the workers job. Thus, the chance that any one of the six supporting sensors is down is 0.85^6 or 37%.

The dynamic architecture however, provides a significant advantage in a lossy wireless environment. Since the fusion schemes outlined above do not require all sensors to be functioning at all times a decision can be made when even a single sensor remains. In fact, all sensors must be unavailable before a classification will be missed. Thus if 1 packet is lost, the system continues with data from the other N-1 sensors. The probability that all 6 sensors are down in this system is $(1 - 0.15^6)$, yielding 99.99% reliability.

6.2 Activity Space Coverage

We want to identify the the classification performance of sensor locations for each activity, not because we want to identify best performing single sensors but because we want to identify a set of sensors offering redundancy on important activities and complementary coverage on all activities. The visualization in Figure 6.1 shows the F-Measure (also called F1) of each sensor for each activity. F-measure is used because accuracy can only be calculated as an overall measure, for specific activities some combination of that classes recall and precisions must be used. For this application precision and recall are weighed equally, meaning we do not favor one over the other. Precision, recall and F1 on a per class basis are defined as:

$$f1 = (2) * \frac{precision * recall}{precision + recall} \tag{6.1}$$

$$precision = \frac{true\_positive}{true\_positive + false\_positive} \tag{6.2}$$

$$recall = \frac{true\_positive}{true\_positive + false\_negative} \tag{6.3}$$

If a user is interested in identifying an activity with ergonomic consequences, such as lifting (shown with a dotted line in Figure 6.1) then certain conclusions can be drawn regarding sensor selection.

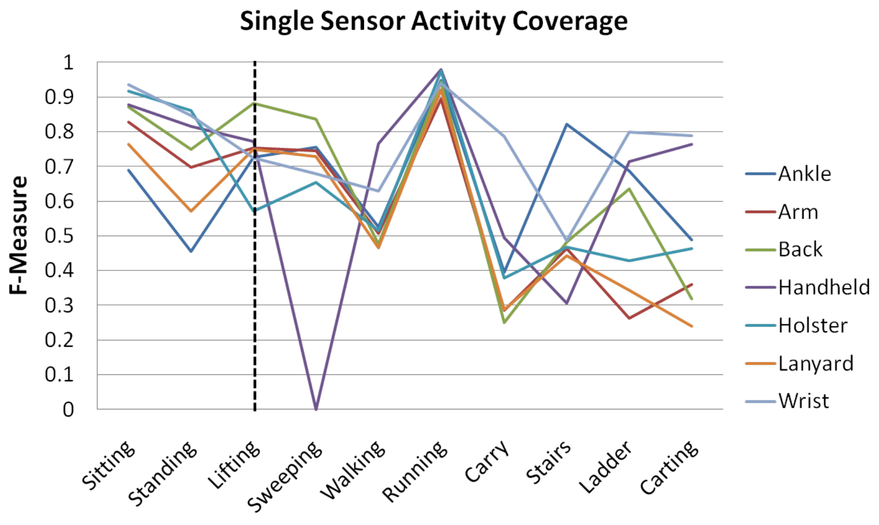

Figure 6.1: A visualization of the performance of sensor locations as they vary with activity.

6.3 Classification

To evaluate classification performance we consider three metrics. The accuracy of classification, the memory footprint of the model, and the classification delay. The accuracy and decision latency are metrics that will impact a humans use of the system and memory footprint is a metric critical to sensor device selection. For completeness and to identify the highest performing algorithms for our application we first run a preliminary test to identify configurations requiring further exploration. Since many of the algorithms take significant amount of computation time to generate some algorithms did not finish, indicated by a DNF label. The preliminary experiment was performed using the dynamic architecture as opposed to the centralized architecture. The accuracies shown here are an average across all subjects.

Fusion Accuracy		**Window Sizes**		
		1 Seconds	2 Seconds	4 Seconds
Algorithms	KNN	74.0%	77.7%	81.4%
	Decision Tree	73.4%	77.2%	79.1%
	Perceptrons	~65%	74.4%	70.2%
	Boosting	DNF	DNF	81.7%
	SVM	DNF	DNF	~60%

Table 6.1: Preliminary Exploration of Machine Learning Algorithms

Preliminary accuracy results shown in Table 6.1indicated that further analysis would not be performed on Perceptrons or SVMs. It did indicate that tree algorithms, Decision Trees and Boosting, performed well and continued exploration should be considered. For this reason Random Forests, a tree based algorithm, is included in future analysis. The preliminary test also indicated that KNNs performed well. Performance also increased as window size was increased. To account for this one and two second window lengths were dropped and a six second window was added for comparison even though the dataset size was not large enough to ultimately select it for implementation.

6.3.1 Accuracy

Follow up, and ultimately final, experiments included both a centralized and dynamic architecture performance. There was no plan to select a centralized architecture for final implementation, but since it would provide the highest achieveable accuracy it was included. Table 6.2 and Table 6.3 show the performance of the limited algorithms set for both four and six second windows. The reasoning for this algorithm and window size selection is described above.

All Sensors Accuracy		**Window Sizes**	
		4 Seconds	6 Seconds
Algorithms	Clustered KNN	87.2%	89.6%
	Decision Tree	82.6%	85.4%
	Boosting	90.8%	91.2%
	Random Forest	93.3%	93.9%

Table 6.2: Accuracy Results using the Centralized Architecture

Fusion Scheme Accuracy		**Window Sizes**	
		4 Seconds	6 Seconds
Algorithms	Clustered KNN	85.1%	86.8%
	Decision Tree	86.3%	86.1%
	Boosting	88.8%	88.0%
	Random Forest	91.9%	92.7%

Table 6.3: Accuracy Results using the Dynamic Architecture

Considering the table of fusion performance values one may notice that the numbers have increased from our preliminary run for the algorithms present in both. In the case of KNN this is explained by the fact that we added an extra step to both decrease the model footprint and increase performance. Clustering training instances by finding the mean those instances and then replacing the original instances with that mean reduced the number of points stored in the model and also increased generalization, ultimately increasing accuracy performance. Decision Trees, and boosting performance increased because the sweep interval was increased and the step size was decreased. This allowed the parameter optimization process to try additional configurations, increasing performance significantly. Again, the accuracies shown are an average across all subjects.

6.3.2 Memory Footprint

Simultaneous to accuracy calculation was memory footprint calculation. The memory footprint comparison of the final considered models is described in Table 6.4. Since boosting was limited to using 10 trees and random forests were limited to uses 60-150 simpler trees expected scaling would be 10X and 150X memory increase respectively. The unexpected scaling probably due to weka storage format, could be lowered with a custom model storage format. The values show are normalized since the models were generated on the PC and not realistic in terms of size when implemented on the end device.

Normalized Memory Usage		
Algorithms	Clustered KNN	3.9
	Decision Tree	1.0
	Boosting	18.5
	Random Forest	457.4

Table 6.4: Memory usage of different models normalized by the size of a single decision tree. Indicates relative size of different models.

An important and fairly intuitive result with this experiment is that accuracy performance costs in memory. For each percentage of accuracy one can expect an increase of 10X in memory, and even higher than that in the case of random forests. Also referring back to Table 6.3 indicates that KNN and decision tree algorithms offered similar accuracy performance but here we see that KNNs required four times as much memory. This strengthens the case to eliminate KNNs from the optimal algorithms.

6.3.3 Classification Delay

The classification delay is simply a measure of the delay the classifier alone introduces. This is compounded by any delay in the feature extraction methods and any delay introduced by choosing longer windows. The results here again scale with the size of the algorithm, but much less so as all are in the same order of magnitude. Table 6.5 illustrates the execution time. For reference this experiment was performed on a PC and 1 = 300 us.

Normalized Classification Delay Utilization		
Algorithms	Clustered KNN	1.0
	Decision Tree	1.3
	Boosting	1.7
	Random Forest	6.1

Table 6.5: Classification Delay using multiple models normalized by the delay for for a clustered KNN decision

No significant information was obtained from this analysis warranting any change in algorithm selection. The conclusion for general algorithm perfomance is that tree algorithms work well for this dataset in the dynamic configuration. Memory contraint is the biggest issue for a designer and the corresponding tree algorithm should be selected based on device memory availability. For this reason, additional experiments are performed using random forest decision algorithms on the MC9500 devices and boosting algorithms on the eWatches.

6.4 Fusion Methodology

As described in Section 5.4.2 there are numerous methods of combining sensor confidence data. The results here are obtained by generating models for all seven sensor locations and then combining them via fusion. We compare all the fusion methodologies in parallel with the accuracy of the systems, so that a fair weight is given to all fusion techniques for each machine learning algorithm. In general the best performance was obtained using an average of class proabilities. To account for memory contraints and simulate expected performance on the deployable devices, the results in Table 6.6 are calculated using random forest decision algorithms on the MC9500 devices and boosting algorithms on the eWatches.

Method	Accuracy
Average	89%
Product	59%
Straight Vote	84%
Max Probability	79%
Rank	88%

Table 6.6: Fusion Schemes Compared

6.5 Resilience

In order to evaluate the resilience of the classifier to packet loss faults we simulated random packet loss from 10% to 90%. To evaluate resilience to sensor node failure, we averaged the accuracy of every permutation of subsets size N, where N was 1-7, subject to the constraint that the handheld was always a part of the set. To visualize the difference in reliability the low bandwidth scheme was also included.

Figure 6.2 illustrates the results from this experiment. The primary result is that in all non-ideal environments, greater than 10% packet loss, better classification performance was achieved with dynamic fusion compared to a static low bandwidth classifier. For example, there is a 35% accuracy increase in an environment with 50% packet loss. This is also true for lost sensors, at 10% packet loss a dynamic fusion scheme may lose up to three sensors and still outperform a static fusion scheme with a full sensor set. In an environment without loss we have discovered that a dynamic fusion scheme classifies with only 2% lower accuracy than the low bandwidth scheme. Again, this experiment was performend using random forest decision algorithms on the MC9500 devices and boosting algorithms on the eWatches.

We have also discovered that the number of sensors can be reduced in low loss environments for bandwidth savings. Three sensors is an effective configuration in terms of bandwidth utilization, accuracy, and wearability of the system.

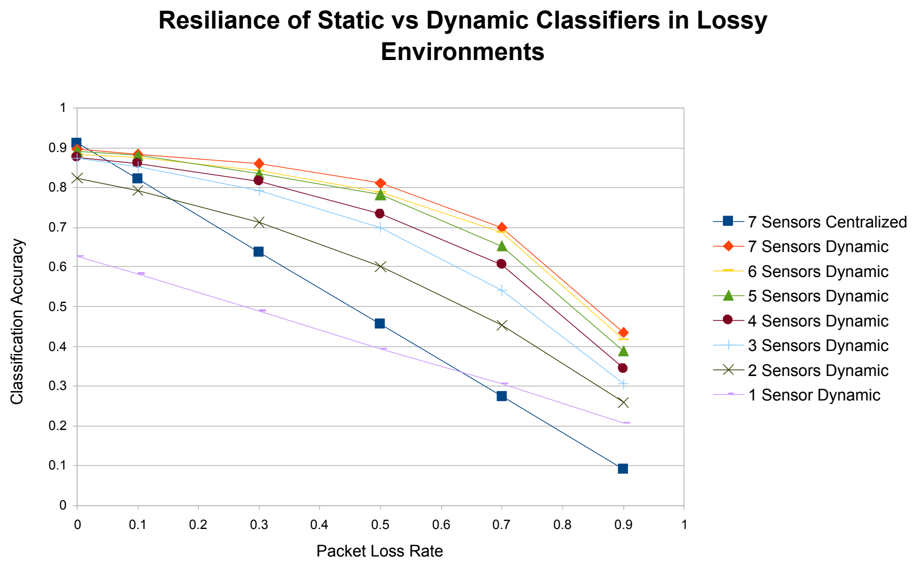

Figure 6.2: With the same number of sensors in a 0% loss environment the dynamic scheme yields 2% lower accuracy than a low bandwidth scheme. With any amount of packet lost and up to three lost sensors, the dynamic scheme performs better than the low bandwidth scheme.

6.6 Energy Savings

The methodology for energy experimentation as descrined in the Section 5.5.2 was evaluated. As expected, accuracy decreased as penalty time per incorrect classification was increased. It was discovered that diminishing returns on accuracy did not occur at the same penalty time point for all sensor configurations, but increased in time with the number of sensors. Intuitively this makes sense because with more sensors less information is lost when a single sensor is put to sleep. Figure 6.3 illustrates the effect of penalty time on accuracy which is bounded by the 60% accuracy of the always on master device. The point of diminishing returns is shown with a horizontal cutoff at 63.3% accuracy with a standard deviation of 0.5%. This corresponds to 30-60 second penalty times depending on the sensor configuration.

Power analysis revealed that higher sleep times offered large power savings, however diminishing returns on power, occurred at 14-16 seconds across all sensor configurations. A visualization of the power optimized data is shown in Figure 6.4, with the vertical dotted line representing the point of diminishing returns. Since power savings hit a point of diminishing returns before accuracy loss, a shorter sleep penalty time would be best for most applications.

Accuracy performance is lower in this experiment due to the lower fidelity of scenario data, no personalization was performed in this experiment. The four subject scripted dataset was collected to emulate in the wild data and boosting was used for all of the models.

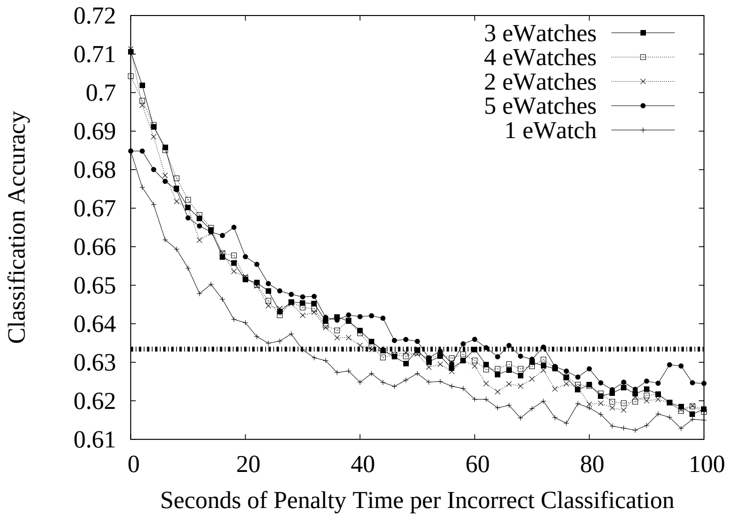

Figure 6.3: Accuracy vs Penalty Time with a handheld master. The dotted line depicts the point of diminishing returns.

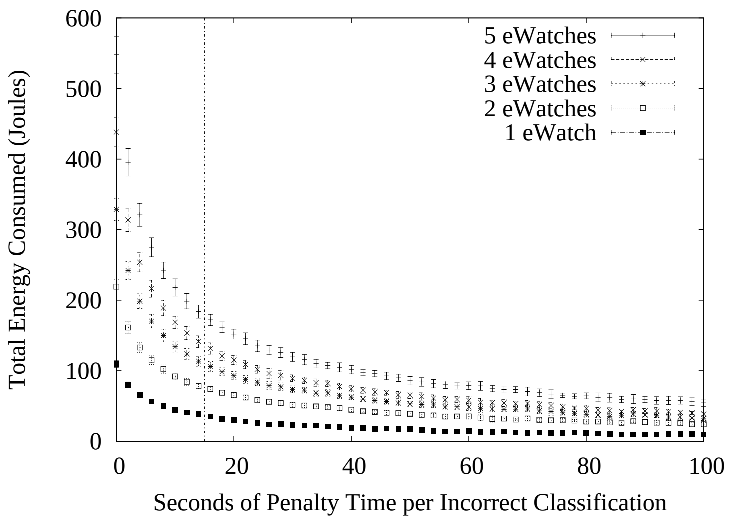

Figure 6.4: Energy vs Penalty Time with handheld master. The dotted line depicts the point of diminishing returns.

6.7 Tradeoffs

Figure 6.5 illustrates the implications of the three architectures. The metrics considered are wireless bandwidth, accuracy, and energy. The energy shown here is eWatch operating energy requirements since no other type was considered. This is the max energy at the given configuration. If one considers the penalty scheme this energy would decrease significantly as shown in Figure 6.4. Radio power would also scale with bandwidth utilization further lowering energy consumption in dynamic configurations.

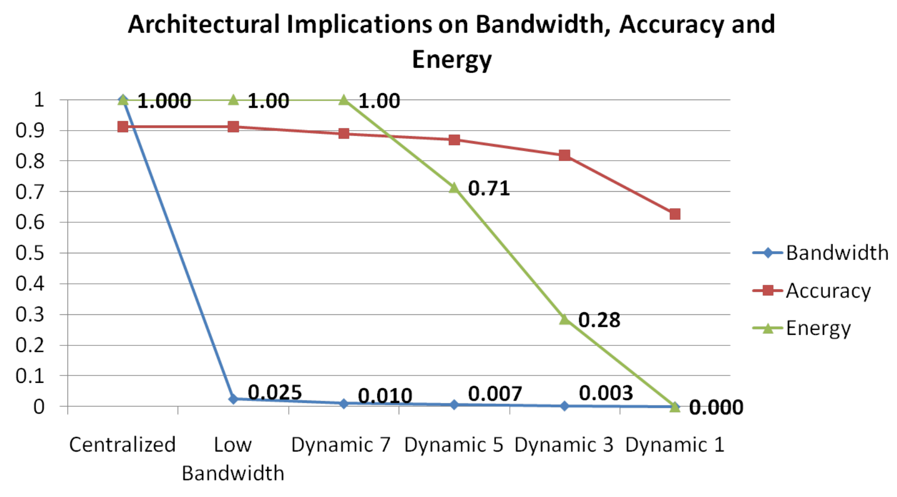

Figure 6.5: Wireless bandwidth, accuracy and power comparison for each type of architecture. The Y axis is normalized for bandwidth with 1.0=12KB/sec. Centralized and Low Bandwidth both utilize 7 sensors. Dynamic 1 utilizes 0 bandwidth because it is local master decisions only.

6.8 Customer Scenario

As a deployment example and to demonstrate the flexibility of the system we describe here a customer scenario. Many different scenarios could exist so we choose a likely one. In this scenario the customer is primarily interested in identifying lifting events. This customer claims that they have a single MC9500 that their employee carries and two supporting eWatches. Upon consulting with Figure 6.1 we identify the Back, Handheld and Arm as the top three performers in identifying lifting. The customer would definitely want to include a lower back sensor, perhaps in a lifting belt, a handheld device for redundancy in lifting detection, and a wrist or ankle sensor to complement the back sensor and capture other activities.

Next the customer would choose the deployment algorithm depending on the computation and memory constraints of the device. In this example our customer has an MC9500 in the handheld location and eWatches for both the back and wrist positions. The MC9500 is a computationally powerful device and permits the use of Random Forests while the eWatch is memory constrained thus requiring the customer to drop to Adaboosting. Since dynamic sensor fusion is used it does not matter that different algorithms are used, only that confidence information is available, which is the case for all machine learning implementations considered in this study. A quick simulation of the customer's setup provides the confusion matrix in Table 6.7, indicating recall and precision of 88.8% and 94% respectively for lifting and high performance on most other activities.

Sitting	Standing	Lifting	Sweeping Mopping	Walking	Running	Carry	Stairs	Ladder	Carting		Recall	Precision
96%	2%	0%	0%	0%	0%	0%	0%	0%	1%	Sitting	0.956	0.988
2%	**92%**	0%	4%	0%	0%	0%	0%	0%	1%	Standing	0.925	0.938
0%	0%	**89%**	9%	1%	0%	2%	0%	0%	0%	Lifting	0.888	0.940
0%	1%	3%	**89%**	1%	0%	1%	1%	3%	1%	Sweeping	0.893	0.852
0%	0%	0%	1%	**78%**	0%	7%	11%	0%	3%	Walking	0.782	0.810
0%	0%	0%	1%	0%	**98%**	0%	1%	0%	0%	Running	0.975	0.986
0%	0%	0%	6%	7%	0%	**84%**	2%	0%	1%	Carry	0.840	0.686
0%	0%	2%	1%	26%	1%	12%	**57%**	0%	0%	Stairs	0.569	0.625
0%	0%	0%	9%	0%	0%	0%	0%	**90%**	1%	Ladder	0.902	0.893
0%	0%	0%	1%	2%	0%	1%	4%	0%	**92%**	Carting	0.921	0.866

Table 6.7: Algorithm Performance Given Customer Constraints

6.9 Personalization

6.9.1 Scenario without Personalization

The previous results presented do not account for personalization. They identify expected lab performance in a clean environment. By using no personalization the overall accuracy dropped to 61% in a real world scenario. This is a decline of approximately 24% from the cross validated results. Upon inspection of the confusion matrix in Table 6.8 it is immediately that the primary issue is lifting samples being misclassified as sweeping. Note that because the scenario data used a subset of the lab data the matrix is not square. The diagonals are bolded for clarity. This is because in the lab training phase all forms of lifting were not considered, specifically side to side lifting. In the scenario, where boxes are being moved on and off of carts there is a significant amount of side to side motion. Personalization can address some of these training oversights and provide an increase in accuracy. It should also be noted that there were a significant number of errors classifying carting as walking and sweeping. The confusion between using a cart, walking and sweeping varied between subjects, and is a situation where personalization should increase performance.

It is discouraging that lifting has a low recognition accuracy because it has the most potential for corrective feedback. Since ErgoBuddy is designed to identify incorrect lifting trends over time, and provide timely and relevant feedback, classifications should be as accurate as possible. Because the system will likely be providing incorrect feedback initially an opportunity arises for the user to intervene and provide the ground truth required for personalization.

Table 6.8: Scenario Confusion Matrix

Activity	Sit	Stand	Lift	Walk	Carry	Stair	Cart
Sitting	**247**	0	0	0	0	0	7
Standing	0	**391**	1	2	1	0	10
Lifting	0	0	**98**	1	7	0	9
Sweeping	0	17	320	15	4	1	106
Walking	0	0	8	**229**	6	34	153
Running	0	0	0	0	0	0	0
Carrying	0	0	10	49	**181**	11	25
Stairs	0	0	8	13	0	**61**	5
Ladder	1	1	58	4	2	3	67
Cart	0	0	3	2	0	0	**313**

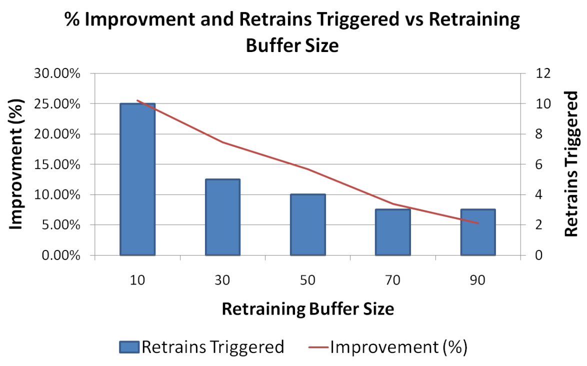

Figure 6.6: Accuracy improvement as a function of retraining buffer size

6.9.2 Scenario with Personalization

In our experiments, ground truth is assumed. However, in a wild application this will not be the case, therefore what we present is an upper bound for accuracy increase using personalization. There are opportunities to obtain ground truth, especially when the system is continually offering incorrect feedback, so we expect that real world performance would be somewhere between performance without feedback and the upper bound presented here.

Figure 6.6 indicates that smaller buffers, and thus faster retraining offers the best performance increase, especially in our relatively short scenario. Given hours of continuous data and sparse field generated labels this phenomenon may not be as pronounced. The best improvement attained was 25% when a retraining buffer size of 10 was used triggering the retraining process 10 times during the scenario. For some subjects scenario personalization performance exceeds the performance of the cross validated lab data experiment. Larger retraining buffers are shown to illustrate that lower frequency retraining will yield worse improvement, and to address the consideration that larger buffers should be used when the algorithm requires a computationally expensive training stage. However, this is not an issue with the kNN algorithm.

Figure 6.7 indicates the before and after realtime accuracy performance using personalization for four subjects. The retraining buffer size for this experiment was 10 to illustrate the highest

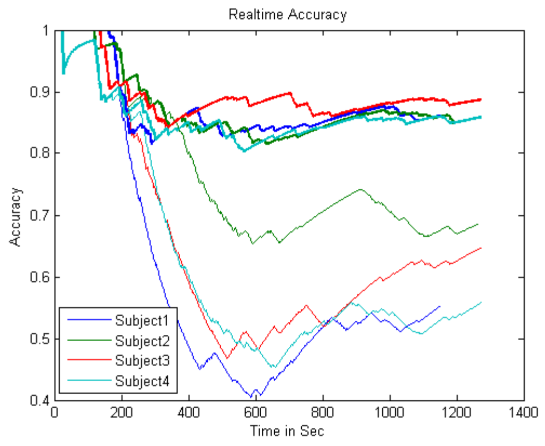

Figure 6.7: Realtime accuracy before and after personalization. Bolded lines indicate higher performance after personalization.

accuracy obtained. The bolded lines indicate the performance over time with personalization. One can observe that the drop in accuracy during the lifting stage of the scenario is largely avoided using personalization.

Section 7

Conclusion

7.1 Contributions

In this report, a wireless, wearable system utilizing sensor fusion was developed for use in activity recognition. A dynamic architecture was developed using confidence based sensor fusion. Resilience to faults and power savings are demonstrated as a result of the flexibility of the architecture. Machine learning is used to obtain the highest accuracy and flexibility of the activity recognition engine. A data collection application was developed on the eWatch and Motorola MC9500 platforms as well as local context recognition engines.

The research questions answered in thus study were:

- Is it possible to accurately detect package delivery activities?
- Can sensor fusion be used to dynamically add and subtract sensors at run-time?
- Does the dynamic nature of the system provide increased resilience to sensor faults and possible power savings?
- Can personalization recover lost performance in the transition from lab to scenario data?

We show that accuracy of up to 93% is attainable using a static sensor set and up to 92% with a dynamically fused set when trained on typical pakcage delivery activities. This utilizes a training and testing methodology which excludes personalized data in both training and testing. However most of the analysis is performed on data that is idealized and collected in a lab setting. We would expect real world accuracy to decrease, but personalization to offset that loss.

We introduce the concept of using confidence based fusion techniques for activity recognition fusion. A comparison of centralized, low bandwidth, and dynamic fusion sensor architectures is provided. Also accuracy is provided as a function of energy, number of sensors and packet loss. Considering the resiliency and power advantages of a dynamic architecture a designer can design the ultimate system around his own constraints. An example of such a design process was described in section 6. The dynamic architecture combined with sensor fusion is able to achieve within 2% accuracy of a static system. Thus we belive that the advantages in regards to resilience and power far outweigh the accuracy loss.

The dynamic nature, allowed for a high level of performance even with sensor loss. In all non-ideal environments with greater than 10% packet loss, better classification performance was achieved with dynamic fusion compared to a static low bandwidth classifier. A power savings simulation by dynamically turning off inaccurate sensors has also been performed. Application requirements can dictate the minimum required accuracy and the maximum desired power consumption of the system, but based on experimental findings, penalty times of greater than 16 seconds are not recommended.

A comparison of lab training, scenario and scenario personalization experiments is performed. The primary goal of personalization is to increase the relevance and accuracy of feedback so that a user of Ergobuddy can reduce injury.

Since our system had initially poor scenario performance and because our system can be worn for long periods of time, personalization was considered. We have quantified the performance of the system as retraining occurs and see encouraging results. We acknowledge that the results are idealized since it is not feasible to have ground truth for every sample when a system is truly deployed. Thus, the results shown here are an upper bound of expected performance increase. However, with user feedback we believe that accuracy will increase compared to a system utilizing a generalized model.

Preliminary work has also been done on the implementation of the dynamic architecture on both the eWatches and MC9500 platforms. The deployment of the system will include the ability to log contexts, the ability to provide realtime feedback, and the ability to provide end of day analytics. This work is of great interest to Motorola and a technology transfer is currently taking place. We strongly believe that the system could be used as a starting point for a dynamic system deployed in a package delivery setting.

7.2 Future Work

As hinted at in the body there are numerous ideas outside the scope of this report that could improve the system.

7.2.1 Implementation Issues

Currently all datapaths necessary to implement a dynamic architecture that allows retraining do not exist. Thus the ability for the system to retrain with new user data or with new activities is prevented. Since the scope of this project did not specifically include retraining, no major effort was not put towards this feature.

Currently the sensor fuser assumes a timeout occurs when data is not received by a certain time period. Potential confusion could occur if a sensor consistently sends decisions past that window. More implementation work could identify latency of each sensor and include this information in the fuser.

7.2.2 Further Research Questions

Since we are working with time series data a HMM could ideally be generated to model the transitions from various activities. This is not feasible with tightly controlled lab data, but if natural data were to be collected this would prove to be an interesting experiment.

A more developed and intelligent process of scheduling sensor downtime would provide better power savings and possibly distribute the load of recognition better. Currently, when a sensor is put to sleep there is no way to wake the sensor until its local timer has expired. A method of keeping the sensors in a low power idle state that is ready to begin recognition upon command would offer more flexibility.

We would like to explore user-to-system feedback mechanisms. Since Ergobuddy is designed to provide audible feedback when a user is performing unsafe activities we hope to design a quick and easy mechanism for the user to confirm or deny that the advice was accurate. This would provide ground truth labels.

Additional work should also be done to conserve power on the master device. Currently this device classifies at all times. The device is interacting with the user so it will need to remain in a ready state, but the classification engine on the device should have the option to be turned off.

References

[1] "Bureau of Labor and Statistics," *U.S. Department of Labor*, 2007.

[2] "Workers Compensation for Production, transportation, and material moving occupations," *Bureau of Labor and Statistics, National Compensation Survey*, 2009.

[3] L. Bao and S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing*, pp. 1–17, 2004.

[4] N. Bharatula, P. Lukowicz, and G. Tröster, "Functionality-power-packaging considerations in context aware wearable systems," *Personal and Ubiquitous Computing*, vol. 12, no. 2, pp. 123–141, 2008.

[5] B. French, D. Siewiorek, A. Smailagic, and M. Deisher, "Selective Sampling Strategies to Conserve Power in Context Aware Devices," in *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, 2007, pp. 77–80.

[6] J. Boyd, H. Sundaram, and A. Shrivastava, "Power-Accuracy Tradeoffs in Human Activity Transition Detection," in *Proceedings of the 2010 Design, Automation and Test in Europe Conference*, 2010, pp. 1524–1530.

[7] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster, "Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*. IEEE, 2008, pp. 281–286.

[8] M. Abidi and R. Gonzalez, *Data fusion in robotics and machine intelligence*. Academic Press Professional, Inc. San Diego, CA, USA, 1992.

[9] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 2002.

[10] K. Van Laerhoven, A. Schmidt, and H. Gellersen, "Multi-sensor context aware clothing," in *ISWC 2002*. IEEE, 2003, pp. 49–56.

[11] T. Stiefmeier, D. Roggen, G. Tröster, G. Ogris, and P. Lukowicz, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing*, vol. 7, no. 2, p. 42, 2008.

[12] Y. Shi, M. H. Nguyen, P. Blitz, B. French, S. Fisk, F. De la Torre, A. Smailagic, D. P. Siewiorek, M. al' Absi, E. Ertin, T. Kamarck, and S. Kumar, "Personalized stress detection from physiological measurements," *International Symposium on Quality of Life Technology*, 2010.

[13] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher, "Activity recognition and monitor-

ing using multiple sensors on different body positions," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on.* IEEE, 2006, p. 4.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[15] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16] (2009) Fedex to adopt rugged handhelds from motorola. [Online]. Available: http://www.computerworld.com/s/article/9138071/FedEx_to_adopt_rugged_handhelds_from_Motorola

[17] U. Maurer, A. Rowe, A. Smailagic, and D. Siewiorek, "eWatch: A Wearable Sensor and Notification Platform," in *Proceedings of the 2006 International Workshop on Wearable and Implantable Body Sensor Networks*, 2006, p. 145.

[18] G. Terrell and D. Scott, "Variable kernel density estimation," *The Annals of Statistics*, vol. 20, no. 3, pp. 1236–1265, 1992.

[19] L. Breiman, *Classification and regression trees.* Chapman & Hall/CRC, 1984.

[20] F. Rosenblatt, "The perceptron-a perceiving and recognizing automaton (Technical Report 85-460-1)," *Cornell Aeronautical Laboratory*, 1957.

[21] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory.* Springer, 1995, pp. 23–37.

[22] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] S. Kullback and R. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.

[25] (2011) Tmultilayerperceptron: Designing and using multi-layer perceptrons with root. [Online]. Available: http://www.fynu.ucl.ac.be/users/c.delaere/MLP/

[26] C. Hsu, C. Chang, C. Lin *et al.*, "A practical guide to support vector classification."

[27] R. Polikar, "Ensemble based systems in decision making," *Circuits and Systems Magazine, IEEE*, vol. 6, no. 3, pp. 21–45, 2006.

[28] T. Ho, J. Hull, and S. Srihari, "Decision combination in multiple classifier systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 1, pp. 66–75, 2002.

[29] T. Wang, Y. Han, P. Varshney, and P. Chen, "Distributed fault-tolerant classification in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 724–734, 2005.