Human Motion Identification: An application of a novel automatic event detection framework

Timothy Hnat, Tamim Sookoor, and Kamin Whitehouse
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{hnat,sookoor, whitehouse}@cs.virginia.edu

ABSTRACT

In this paper, we present a framework for generalized event detection in wireless sensor networks. We claim that a large class of wireless sensor network applications involve event detection, yet many different events can be detected using a few primitive features. We describe an initial implementation of the framework in Matlab and present initial results of our framework by utilizing it to detect human motion. We utilize the framework to differentiate between sitting, standing, walking, and falling using data collected through an accelerometer. This application is motivated by it being useful in an elderly monitoring system such as AlarmNet. Finally, we describe some of the challenges faced in implementing an event detection framework that will be useful for a sufficiently large subset of wireless sensor network applications as well as some of the limitations that such a system possesses.

1. INTRODUCTION

Many wireless sensor networks deployed so far rely on detecting events based on sensor readings. For instance, the Great Duck Island habitat monitoring application [2] detected the presence of storm petrels in burrows, AlarmNet detects events in assisted living facilities, VigilNet [1] detected the presence of enemy combatants and tanks, and PEG [3] detected the presence of the evader in the pursuer-evader game. While the applications that rely on event detection can span all the way from environment monitoring and habitat monitoring to elderly monitoring and battlefield surveillance, the data which are used to identify the events come from a common set of sensors. Most applications make use of a few basic sensors such as light, temperature, humidity, and acceleration, yet an event detection algorithm has to be customized for each application. This can be time consuming and a waste of resources if a common event detection framework could identify events in many different applications. The event detection framework would take as input the streams of sensor readings collected by the sensor network and output a stream of events.

This description of the event detection framework as converting an input stream of raw sensor reading to a stream of events highlights another advantage of such a framework. It filters raw sensor data. This is a very useful feature for wireless sensor networks because the communication bandwidth is a heavily utilized, yet very limited resource. Also, transmitting messages through the radio is one of the most power consuming tasks for a wireless device. Thus minimizing the amount of data that needs to be transmitted over the communication channel is advantageous for sensor networks. Currently, a number of techniques have been proposed to minimize the amount of data that has to be transmitted out of sensor networks. These methods range from filtering the data to aggregating it within the network. Yet, these techniques have a number of weaknesses. Filters are not very accurate and may result in useful information being filtered out if they are not chosen correctly or the sensed data or environmental conditions change during the deployment. In network data aggregation is very application dependent and may not be feasible for certain applications. For instance, if the application involves finding the median of all sensor readings, aggregation would not be applicable. A general event detection framework, on the other hand, would not have a hard threshold like in filtering and would not merely eliminate data in order to minimize the amount of data. It would, instead, consider the available data and output a message which would consist of an event label and a time stamp whenever the input data suggests an event. Also, a framework for general event detection would be much easily utilized for many applications as opposed to in network aggregation techniques which would have to be customized to the applications.

While an automatic feature detection framework would make sensor network application implementation easier for the sensor network experts, it would also bring sensor networks closer to the average person. There is a widely talked about Radio Shack model for sensor networks, where it is envisioned that a homeowner, for instance, could go to a supermarket or electronics store, like Radio Shack and purchase sensors. The homeowner would then deploy the sensors around the area s/he wants to monitor and, with very little effort, implement a useful sensor network application. The state of the art does not permit such a vision yet as the deployment of sensor networks is a tedious task even for a group of researchers with considerable experience. An automatic event detection framework would help bring this vision one step closer to reality. Since the framework is not
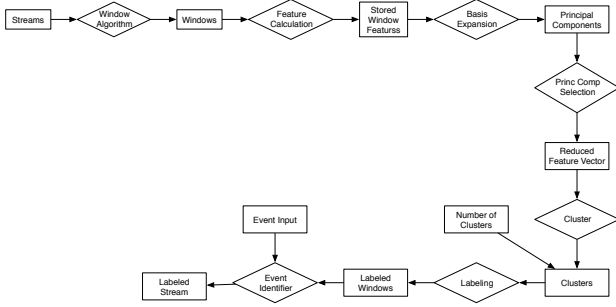
Figure 1: System Flow Chart

application specific, it could be utilized for a wide variety of applications that end users may desire. Also, since it would require very little user interaction, it would make sensor network deployment for event detection applications easier for a layperson.

While we have motivated the general automatic event detection framework in this section, the rest of the paper would focus on the utilization of the framework for the motion identification application. By describing how human motion is identified using the framework, we hope to enlighten the reader on the workings of the automatic event detection framework. Section 2 describes the human motion identification system. We describe the hardware utilized as well as the algorithm, which is essentially the automatic event detection framework. Section 3 provides an evaluation of the system. We describe the performance of the system as various parameters are altered and compare the automatic event detection, or unsupervised learning algorithm, against a manually tuned motion identification algorithm.

2. SYSTEM PROPOSAL

Body monitoring is the goal of our system. Walking, standing, sitting, and most importantly, falling, are events which we are able to distinguish. Traditional techniques use either a manually created filter or some type of training data combined with supervised learning techniques. Our system takes this one step further to include unsupervised learning as an equivalent technique to identify the actions.

2.1 Hardware

A Micaz test platform, combined with the standard sensor board, was used to collect data from various people. The dual-axis accelerometer on the sensor board is limited to 50Hz sampling by the hardware configuration. Sampling is done at 200Hz to avoid any problems with Nyquist limits. Another Micaz was connected directly to a programming board and acted as the intermediary between the sensors and the logging software on the computer.

2.2 Algorithm

Stage one is the transformation of the raw data from the accelerometer attached to a person into segmented slices that accurately represent the original data set. The user must specify how the windows will overlap and how large each window should be. Windows should be chosen so that they can accurately represent the events from the system. For example, one second windows are a great choice for walk-

ing, sitting, and standing, but not be exactly what one needs for falling. Falling might need to examine windows of much smaller sizes. Once windows sizes have been determined, the data is split up into the various windows and a feature vector is computed for each window.

Feature vectors are a critical component of this system. They allow a clustering algorithm to make the groupings that will ultimately be associated with individual actions. We have implemented a small number of features that are sufficient for our applications and we suspect they will be sufficient for many other types of activities that are monitored with the same sensors we use.

Mean and variance are two basic features of data that are really useful for classification. By choosing a window size that is large enough, a fast fourier transform (FFT) can be computed and the individual frequency components identified. A spike or outlier detector is the final feature we are computing. This simply looks for data that is outside the normal, plus or minus 3σ. Windows are simply tagged with a binary value based on whether or not they have a spike. Our sample data contains information from two accelerometers and the features are computed individually on both. All of the features computed are placed into a single feature vector that is associated with the original data window. Even with the small number of features we have chosen to implement, one gets a very large feature vector for each window. This vector is much too large to cluster.

Basis expansion and decomposition is a technique that allows large dimensional space to be reduced and ordered into the components that distinguish the most amount of data. Our algorithm uses principal component analysis to accomplish this but any other technique will work. For now, the user must specify how many components to use in the clustering algorithm. It is still an open question to determine, automatically, the number of components that are required for representation.

The final component, and the most important, is the clustering of the reduced feature vectors. At this point, all of the data is grouped into useful classifications that can be turned into events. We have chosen to cluster with a gaussian mixture model (GMM) with expectation maximization. GMMs provide a probabilistic model of clusters which is important because our events will generate many different types of variance and contain differing numbers of data points. The gaussian models will also all an overlap of clusters, with some points being equally likely in both clusters or the clusters can contain another cluster. Overlapping clusters can show similar events or events that have transition edges between the pair. A cluster contain another cluster can show that one activity is a subset of another. Determining the correct number of clusters is an open questions and the user is required to supply the number of events that will occur during monitoring.

Event labeling is the final step, which must be done by the user. The clusters will correspond to distinct activities and the user must label each cluster with its corresponding event. Once a cluster has been labeled, it is possible to associate future occurrences of the event with the appropriate label.

3. RESULTS

In this section, we measure the performance of our algorithm on difference data sets. Our goal is to show that for different people, we can perform better than the manual
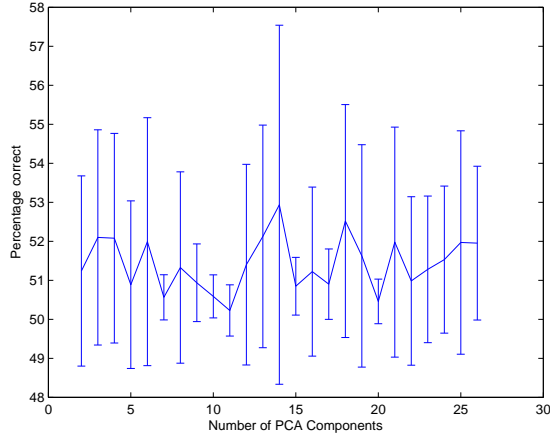
Figure 2: Principal Component Evaluation

Figure 3: Window Size Evaluation

approach.

3.1 Data

Results are based on the data collected in three different locations with two different people. A sensor was attached to the hip on a belt in order to collect the data. Walking, standing, sitting, and falling were the actions performed for the evaluation. A second person watched and recorded the time stamps when each action started or changed. This labeled data provides the ground truth from which all the algorithms are compared against.

3.2 Principal Component Evaluation

Principal component reduction is useful for eliminating the unnecessary components from the evaluation. Because the data can be noisy, eliminating the components has helped the algorithm in some cases. Figure 2 shows that in our test data, removing components does not affect the results. This data is averaged over 100 runs and shows plus and minus one standard deviation. Because we only have a maximum of 28 components in the feature vector, there may not be nearly enough to adversely affect the algorithm. If this feature vector had thousands of components, then PCA would be beneficial. Another reason to use PCA is to reduce the dimensionality of the data that we have to cluster. This will improve performance of the algorithm.

3.3 Window Size Evaluation

Window sizes will affect both the accuracy and the performance of the algorithm. Figure 3 shows the results of varying the window size. Intuitively, the smaller the window size, the more accurate the evaluate. One of the problems with choosing small window sizes is that it hinders the computation of the feature vector. We use an FFT algorithm as part of the vector and without large amounts of data, it is difficult to compute proper FFTs. This figure shows that the smaller the windows, the better the evaluation. With large windows, some of the features of the data get hidden in the computation and are not recoverable. Once the window size reached about 300, (1.5 seconds of data), it leveled off and did not fluctuate very much afterwards. Larger windows are more stable, having less variance, then
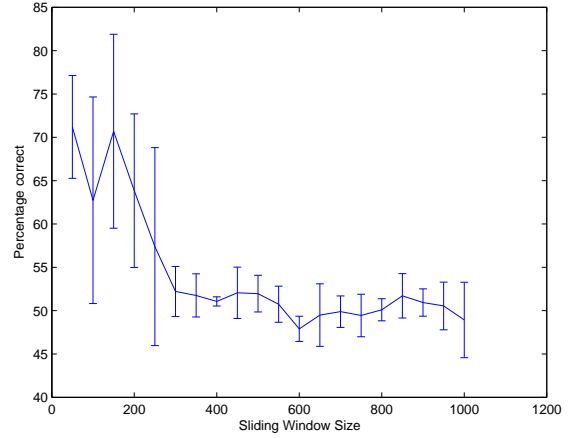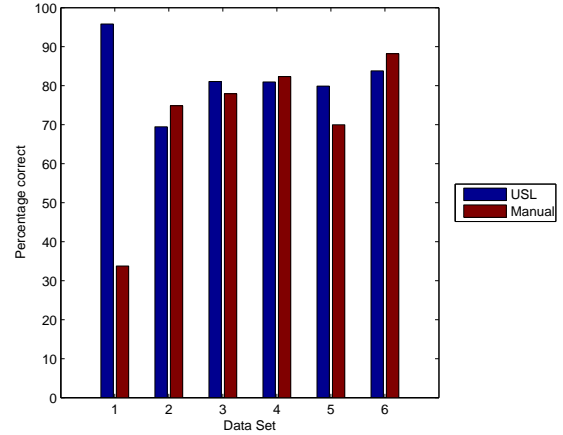

Figure 4: Application to different data sets

the corresponding smaller ones.

3.4 Unsupervised Learning vs. The Manual Method

The key point to this research has been to show that we can do about as well as other techniques. A manual filter was designed to analyze the data using means and variances to determine when a particular event occurs. Because the type of data is simple enough, small number of very distinct events, this filter only took a few hours to design and provides good performance. The filter was designed for one of the data sets and applied to the rest. Figure 4 shows the comparison between unsupervised learning and this manual filter.

Since the data was collected to be similar, the filter should be able to correctly identify events in all of the data. In five out of the size cases, the manual filter performed well. The other case proves a point about manual filtering; It does not work well for situations it is not designed for. Case 1 data had the sensor board attached slightly differently to the body and thus provided data in a different orientation. This figure shows the unsupervised technique is capable of
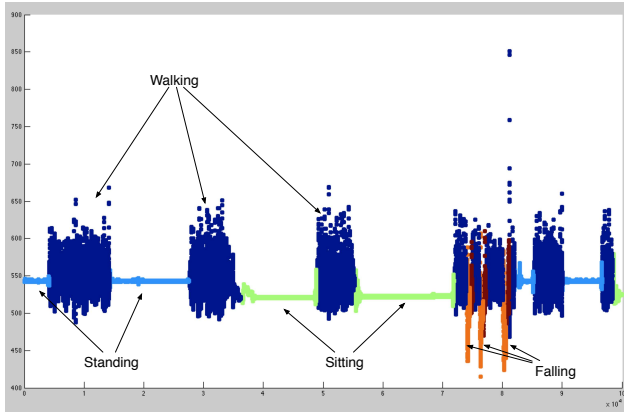
Figure 5: Final labeled output

performing as similarly to a manual method without the problems of retuning the filter for changes in sensor orientation/location.

3.5 Example Output

End results of the algorithm provide a labeled stream of data that can be used to assign real-world events to the data. We show an example run will all of the data labeled by the algorithm. Data is shown in its raw form with the colors representing the different labels. It is very clear with the differences between walking, standing, and sitting. Falling is identified by a short, large spike. This test shows an accuracy of 92.8% , with some of the error can be attributed to problems with the ground truth data (some of the timing might be off a little) and the transition regions will be a source a error.

4. CONCLUSIONS AND FUTURE WORK

In this paper we present a human motion identification system as an application of an automatic event detection framework. We demonstrate that human motion identification is possible using data collected from two accelerometers worn at the hip. The algorithm utilized to identify events is sufficiently general to be used in other event detection applications. This enables the reuse of a single algorithm in multiple applications, reducing the workload in deploying sensor networks.

Another advantage provided by an automatic event detection framework is in network data reduction. By outputting events rather than all of the raw sensor data, an event detection framework would reduce bandwidth utilization considerably. Since bandwidth is a constrained resource in wireless sensor networks and transmitting data is one of the most power hungry activities of sensor motes, an automatic event detection framework would improve the efficiency of sensor networks on two fronts.

The human motion identification system described in this paper has a number of real world applications. The most obvious is elderly monitoring where identifying if an elderly person has fallen could help in getting assistance to such a person in a timely manner. Another application of human motion identification is in a system that monitors activities of daily living (ADL). Such systems have been implemented and are motivated by the fact that people have patterns of activity and changes in these pattens could signal health problems.

While we have demonstrated an application which uses the automatic event detection framework, the framework is still in a preliminary stage. There is considerable work left to be done before it can be used in an actual sensor network. As we have mentioned in numerous locations in this paper, one advantage of the framework is minimizing the amount of data output from the network. This requires two things which are currently not implemented. One is an ability to successfully cluster streaming data. Our current implementation clusters collected data so it has the advantage of "seeing" all of the data while clustering. We have implemented an on-line clustering algorithm, yet it is currently very unreliable and therefore we resorted to a standard clustering algorithm over all the data for this project. The second requirement is porting the framework to sensor motes. This is challenging due to the limited resources on motes. Our current implementation takes a considerable amount of computing resources to execute and, therefore, paring it down to the bare essentials in order to run on motes is a future goal.

5. REFERENCES

[1] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy-efficient surveillance. *ACM Trans. Sen. Netw.*, 2(1):1–38, 2006.

[2] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[3] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, 2005.