Power-Accuracy Tradeoffs in Human Activity Transition Detection

Jeffrey Boyd, Hari Sundaram, Aviral Shrivastava
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, Arizona, USA
{Jeffrey.Boyd, Hari.Sundaram, Aviral.Shrivastava}@asu.edu

***Abstract* — Wearable, mobile computing platforms are envisioned to be used in out-patient monitoring and care. These systems continuously perform signal filtering, transformations, and classification, which are quite compute intensive, and quickly drain the system energy. The design space of these human activity sensors is large and includes the choice of sampling frequency, feature detection algorithm, length of the window of transition detection etc., and all these choices fundamentally trade-off power/performance for accuracy of detection. In this work, we explore this design space, and make several interesting conclusions that can be used as rules of thumb for quick, yet power-efficient designs of such systems. For instance, we find that the x-axis of our signal, which was oriented to be parallel to the forearm, is the most important signal to be monitored, for our set of hand activities. Our experimental results show that by carefully choosing system design parameters, there is considerable (5X) scope of improving the performance/power of the system, for minimal (5%) loss in accuracy.**

I. INTRODUCTION

Human activity detection is becoming increasingly important, not only for high-end athletics training, interactive and immersive games and virtual reality environments, but also in healthcare, both for in-patient training and out-patient monitoring and support. For example, a stroke survivor's physical therapist wants to know if their patient is using their affected arm during the course of the day, how they are using it, (e.g., reaching out or writing), and how many times.

Human activity detection is typically done by attaching position or acceleration sensors on the affected part of the body, and then logging and analyzing the sensor outputs. The analysis consists of several signal filtering, signal transformations, and pattern classification steps, that are quite computationally intensive. In order to provide maximum freedom of movement for the patient, and achieve maximum monitoring, all this computation must be performed on a battery operated mobile device, which the patient has to carry all the time. Given the limited storage capacity and the critical need to minimize the battery weight to carry, it is desirable to implement this patient activity monitoring system in a power-efficient manner. This paper explores the power/performance and accuracy tradeoffs in the design of a human activity detection system.

Our hand-movement monitoring system comprises of a wrist mounted 3-axis accelerometer, and we intend to monitor a set of patient activities, including *sitting*, *standing*, *walking*, *reaching forward* (as if one wanted to pick up an object in front of them) and *lifting the hand* (as if to eat). As opposed to trying to classify sensor output signals into human activity at each moment, we detect the change in the pattern of the signal rather than a change in the signal itself. This scheme is called *Activity Transition Detection*, and has been shown to be more power-efficient. Fundamentally, this scheme has two main steps, feature selection and transition detection, and implementation of an activity transition detection system requires making several choices, including sampling frequency, feature detection algorithm, dimensionality of feature, and width of the window of transition detection etc. The choice of each of the parameters and algorithms essentially trades-off power and performance for accuracy.

While some of the key design parameters, e.g., sampling frequency have been explored by previous researchers [7], previous works have not performed a multi-parameter exploration that we present in this work. Our experimental results underline the importance of design space exploration for designing an accurate, yet power-efficient activity transition detection system. By carefully selecting design parameters and algorithms, and giving a leeway of even 5% on accuracy, we can improve the power/performance by up to 5X. From the results of this *holistic* design space exploration, we cull out several rules of thumb for quick, yet power-efficient design of such systems. For example, we observe that i) significant power/performance can be gained at little loss of accuracy by reducing the dimensionality of feature detection, ii) the x-axis of the output of 3-axis accelerometer, which was oriented to be parallel to the forearm, is the most important signal to be monitored, for our set of hand activities. This is, even though we differentiate between sitting and standing as activities.

The rest of the paper is organized as follows: We start with discussing previous work (Section 2), and then explain our research problem in more detail (Section 3). Section 4 focuses on the design space of our system and Section 5 explains the transition detection method further. Section 6 defines the evaluation metrics for transition detection method. Section 7 discusses experiments results, its analysis and insights gained. Finally, we conclude in Section 8.

This work is funded in part by Science Foundation Arizona (SFAz) and the NSF IGERT (NSF DGE-05-04647).

II. Related Work

Stäger, et al. [7] presented an empirical design methodology to explore the trade-offs between power and accuracy. Their work used a wrist-mounted device with accelerometers and a microphone to capture data of people using kitchen appliances. They took a low-power approach from the beginning, investigating and showing how sampling frequency and feature selection change their system's power consumption. Their work focused on activities where the user interacted with sound-making kitchen appliances whereas the method we describe in this paper does not depend on sound.

Huynh, et al. [2] also use a probabilistic model and sliding windows to detect activity patterns. Though they don't specifically try to detect transitions between activities, they mention their system is capable of detecting transitions. Also, they do not take power into their design considerations.

Krause, et al. [4] used the accelerometers on the eWatch system to classify five activities: walking, running, standing, sitting and climbing or descending stairs. They showed a 4x increase in the lifetime of their wrist-mounted system, without significantly reducing prediction accuracy, by reducing the sampling frequency and exploring different schedules to run their classifier. They refer to these schedules as selective sampling strategies.

French, et al. [1] expanded on the work of [4] by collecting more data and specifically evaluating different selective sampling strategies. The strategies they tested were a baseline uniform sampling strategy, one that samples over the distribution of duration times of activities, and one that samples based on the probability of a transition occurring. Our work in activity transition detection is an alternative to the selective sampling strategies [1] and [4] used. Whereas their methods rely on *a priori* knowledge about the duration of activities and probabilities of transitions, our methods assume no prior knowledge. Further, we explore a much larger design space beyond just sampling rate.

III. Transition Detection Outline

Our goal is develop methods of sensing on small, wearable computing platforms that minimizes power consumption without sacrificing too much performance. To accomplish this we need to first define a set of activities we are interested in, determine what sensors can sense these activities, develop some method to accomplish our stated task of detecting activity transitions, and then systematically explore the independent variables in the system until arriving at a combination that satisfies some design constraint. Fig. 1 outlines the general procedure of our transition detection system. Sensors create a signal representation of a patient's activity. Features are calculated from this signal. The temporal resolution, meaning the sampling frequency and other time-related constructs, affect how the representation of the signal and features, which are then used by our transition detection algorithm.

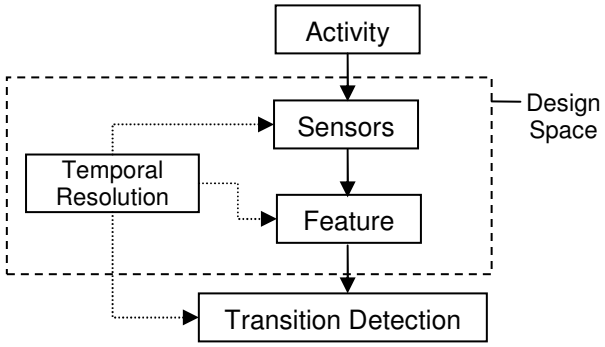

Figure 1. System Flow Chart. The design space in low-power activity transition detections focuses on the sensors that detect activities, the features extracted from these signals, and the temporal resolution or time-dependent properties of the system.

We choose a set of basic activities that physical therapist of stroke survivor wants to monitor. The set of activities are, *sitting*, *standing*, *walking*, *reaching forward* (as if one wanted to pick up an object in front of them) and *lifting the hand* (as if to eat). These activities and gestures are the building blocks to other, more complex activities and gestures, such as the activities of daily living [3], a widely used list of common activities used to assess the function of the elderly or infirm.

We propose to detect transitions using a sliding window technique that compares two blocks of time and computes the likelihood that the activity is different between those two blocks. Several samples are grouped into observations, from which features are calculated. These observations are then grouped into "windows," or blocks of time the algorithm looks at to detect transitions. We next discuss the design space followed by a more thorough description of the transition detection algorithm.

IV. Design Space

In the following sections we outline the parameters of our design space in three broad categories: sensors, features, and temporal resolution, meaning the various time-based controls (sampling frequency, window duration, etc.) we have in the system. We acknowledge they represent just a few options compared to what is possible

A. Sensors

To sense the activities we are interested in we chose to use a wrist-mounted triaxial accelerometer. Accelerometers offer several advantages, beginning with the fact that they're small, lightweight, and inexpensive. Accelerometers are also widely used in the literature on wearable computing systems and human activity recognition [9]. We also investigated using gyroscopes and magnetometers. We found the magnetometers in our test system to be too noisy for any practical use and preliminary tests that included gyroscopes yielded poor results.

We used a triaxial accelerometer and experimented with all seven possible combinations of signals, {x-axis; y-axis; z-axis; x and y axes; x and z axes; y and z axes; x, y, and z axes}.

B. Features

Features are some aspect or quantitative measurement of the signal. They can be simple time-domain measurements such as maximum, minimum, variance, and mean, or frequency-domain based such as the Fast Fourier Transform (FFT) and the Discrete Cosine Transformation (DCT). Other projects have had success using wavelet transformations [5], [6]. Each feature has its own computational complexity, summarized in Table 1. Computational complexity of feature

extraction is important because it is directly related to power consumption [8].

TABLE 1. FEATURES AND COMPUTATIONAL COMPLEXITY. THE COMPUTATIONAL COMPLEXITY AND DIMENSION OF EACH FEATURE AFFECTS POWER CONSUMPTION.

Feature	Computational Complexity
Max	O(N)
Mean	O(N)
Min	O(N)
Variance	O(N)
FFT	O(N log N)
DCT	O(N log N)
Haar Wavelet	O(N)
Daubechies Wavelet	O(N)

C. Temporal Resolution

The third variable is sampling frequency. We chose 100Hz as a baseline. We chose this value because the fastest hand movements are about 5 Hz, and a good rule-of-thumb is to oversample about 20x when using a noisy sensor. Realizing there are low-power advantages to sampling at lower frequencies and encouraged by the good results of Krause et al. [4], who sampled at much lower frequencies, we also experimented sampling at 50, 20, and 10 Hz. Lower sampling frequencies mean fewer samples to process, faster runtimes, and increased power savings.

The fourth variable is the size of the observation described in Sections 3 and 5. We call this observation a frame and define it as the number of samples that the above features are extracted from. In our study we used frame sizes of 10 and 20 samples.

Last we have the length or duration of the sliding window, measured in seconds. The length of the window affects the number of observations used to calculate the likelihood function, as well as the total number of comparisons. In our study we used window lengths of 6, 8, 10, 12, 14, 16, 18, and 20 seconds.

As we've defined them, there are 4480 combinations of these variables in our design space.

V. TRANSITION DETECTION

Our transition detection method uses a measurement of how different two sections of the signal are within a bounded window of time. We split the window in half, creating left and right window panes, as seen in Fig. 2. We want to get some measure of how different the panes are from each other and the entire window, so for each of these panes and for the entire window itself, we calculate a log-likelihood function for each signal we are analyzing:

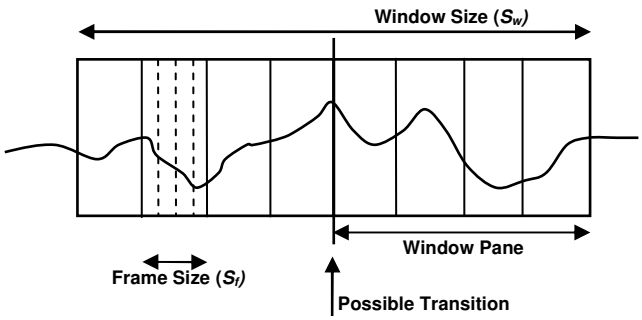

Figure 2. Our sliding window technique. The window is divided into left and right panes and compared using a log-likelihood ratio test. The window panes are made of several frames, which are comprised of several samples (shown in dashed lines). Features are calculated per frame and can be a scalar or a vector.

$$L(x_1, x_2, ...x_N) = \sum_{j=1}^{N} \ln\left(p(x_j \mid \boldsymbol{\mu}, \boldsymbol{\Lambda})\right) \quad (1)$$

where $x_1, \ldots, x_N$ denote the N observations from the left, right or the whole window, $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ are the mean feature vector and covariance matrix in the Gaussian model. The probability p is derived from the multivariate Gaussian probability distribution function. Each observation is a vector of features extracted from each signal.

Once these likelihoods have been calculated for each signal, we combine them in a log-likelihood ratio test, seen in Equation 2.

$$RT_i = \frac{L(i - \frac{N_f}{2} + 1, ..., i + \frac{N_f}{2})}{L(i - \frac{N_f}{2} + 1, ..., i) + L(i + 1, ..., i + \frac{N_f}{2})} \quad (2)$$

In this equation, the variable i represents the frame immediately left of the center line in Fig. 2 labeled "Possible Transition" and N_f is the total number of frames per window. The ratio RT will be close to one when no transition is present and greater than one when a transition is present. It peaks where the probability of a transition is greatest.

VI. EVALUATION METRICS

To evaluate our transition detection system, tested each combination of variables on their accuracy and runtime. The following subsections describe our accuracy measurements and our model for computational complexity, which directly affects runtime.

A. Accuracy

Our accuracy measurements are based on *hits*, *misses* and *false positives*. *Hits* are the number of times the log-likelihood ratio test correctly detected a transition, *false positives* are when it detected a transition when none was there, and *misses* are the number of times it did not detect a transition. We then combine these into *precision*, and *recall*, where

$$\text{Precision} = \frac{\text{Hits}}{\text{Hits} + \text{False Positives}}, \quad (3)$$

and

$$\text{Recall} = \frac{\text{Hits}}{\text{Hits} + \text{Misses}}. \quad (4)$$

A common measure that combines both is the F-Score:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}. \quad (5)$$

F is 1 when both precision and recall are 1. We also a define Reverse F-Score (RF) measure:

$$RF = 1 - F \tag{6}$$

which reverses the F-Score so that 1 is bad and 0 is good. We use RF to more easily visualize accuracy vs. runtime.

B. *Computational Complexity*

In this work, we estimate power as just computation complexity metric, since to a first order, power consumption is very strongly correlated to computational complexity. This is because computational complexity directly affects runtime and runtimes affect power consumption. We've developed a model for computational complexity based on the variables in our system, which we summarize in Table 2.

TABLE 2. DEFINITION OF MODEL VARIABLES. THESE ARE SOME OF THE PARAMETERS IN OUR DESIGN SPACE. D IS A SCALAR FOR *MEAN*, *MIN*, *MAX*, AND *VARIANCE*, BUT A VECTOR FOR *DCT*, *FFT* AND THE TWO WAVELET TRANSFORMATIONS.

Variable	Description
F_s	sampling frequency (samples/sec)
S_w	window size (sec/window)
S_f	frame size (samples/frame)
D	dimension of the feature

The computational complexity (C) of our system can be broken up into two parts: feature extraction (C_{FE}), and the log-likelihood ratio test (C_{RT}). We define:

$$C = C_{FE} + C_{RT} \tag{7}$$

If this were running in realtime, or in other words, the steady-state case, then the best way to look at it is to consider the complexity per comparison. For feature extraction:

$$C_{FE} = F_c(S_f) \tag{8}$$

where F_c is the complexity of the chosen feature. For example, N for max/min/mean and N·log(N) for FFT and DCT. This equation assumes that only one signal and one feature are being analyzed. The second part of complexity is:

$$C_{RT} = \left(\frac{2 \cdot F_s \cdot S_w}{S_f} \right) \cdot D^2 + D^3 \tag{9}$$

where $\left(\frac{2 \cdot F_s \cdot S_w}{S_f} \right)$ is twice the number of frames per window. Multiplying by two is necessary because in each comparison the window is essentially processed twice by looking at the left and right panes and the whole window. The square and cube powers in (9) are there because of the calculation of the covariance matrix and the inverse of the covariance matrix in the multivariate Gaussian probability distribution function. See the Appendix for more details on the derivation of Equation 9.

VII. EXPERIMENTS

For this experiment we used a SparkFun 6-DOF IMU v3 as seen in Fig. 3. The SparkFun device features a Freescale MMA7260Q 3-axis accelerometer as well as gyroscopes and magnetometers, though only the accelerometers were used in this experiment. This device uses a LPC2138 ARM7 microcontroller and Bluetooth to communicate with a computer. The SparkFun IMU was rigidly mounted to a subject's right wrist while they performed sequence of activities. The device was mounted such that the accelerometer's x-axis was parallel to the forearm, pointing toward the elbow, the y-axis perpendicular to the forearm, pointing in the same direction as the thumb when it is outstretched, and the z-axis pointing into the hand from the back of the hand to the palm. All data was sampled at 100Hz and processing of the data was done off-line using Matlab.

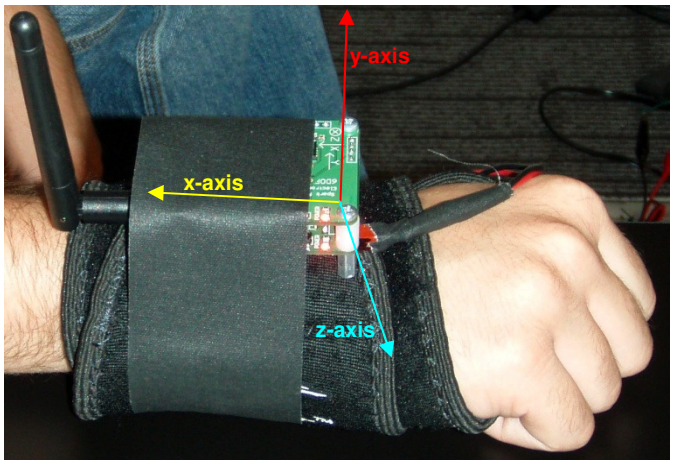

Figure 3. The SparkFun 6-DOF IMU v3. This device features a 3-axis accelerometer, 3-axis gyroscope, 2-axis magnetometer and Bluetooth connectivity. In our tests, this device was attached to the right wrist.

A. *Power/Performance and Accuracy Trade-offs*

We estimate the accuracy and measure the performance of each sequence of activities, for each design alternative. Accuracy is measured as Reverse F-Score (RF), while performance is computed as average runtime per activity sequence. We define runtime in this way because we want to compare the runtimes across all activity sequences, which are of different durations.

Fig. 4 plots the performance and accuracy of all design alternatives. Out of the total 4480 combinations, only 15 are pareto-optimal, and are connected by a curve, and marked by circles on the graph. The pareto-optimal points are also summarized in Table 3. Each pareto-optimal design point represents a design alternative for which there is no better performing design alternative for a given F-Score. These are the most interesting alternatives in the design space.

A couple of interesting observations can be made about these Pareto optimal design point. The difference in accuracy between the top two rows in Table 3 is only 5%, but the top combination runs ~5.6x longer than the second, meaning it's computational complexity and power consumption is much greater. *Thus, significant power savings can be achieved if small sacrifices in accuracy are tolerable.*

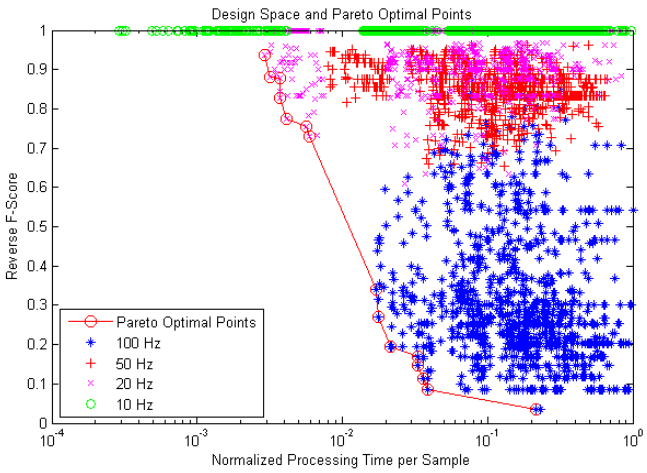

Figure 4. System Design Space. The red circles highlight the Pareto optimal points, which are dominated by the 100Hz and 20Hz sampling frequencies. All combinations with a sampling frequency of 10Hz had an *RF* of 1, meaning they did not detect any transitions. Note the x-axis is log scale and the y-axis is linear.

The eight pareto-optimal points on the lower right side of Fig. 4, all have a sampling frequency of 100Hz, while the seven on the top-middle are all sampled at 20Hz. Interestingly there is not much gain in accuracy when sampling at 50Hz ("+" marked points) as compared to 20 Hz (points marked by "x"), while there is significant improvement in performance. It appears the 10Hz combinations ran very fast, but simply did not have enough data to identify the transitions.

All Pareto points, except the one with the lowest *RF*, have a frame size of 20 samples per frame. The fact that, on average, the 20 samples per frame combinations ran faster the 10 samples per frame combinations was predicted by our model for computational complexity in the equations described in Section 6.2. Frame size is in the denominator in Equation 9; therefore dividing by a larger frame size reduces the overall complexity. This also makes intuitive sense because with a larger frame size there are fewer frames, or observations, per window and features are calculated per frame.

Note also that 12 of the 15 Pareto points use the x-axis, which represents the line parallel to the forearm, from wrist to elbow. The fact that so many of the Pareto optimal points use this axis indicates that it is important for detecting the kinds of activities and transitions between the activities we tested.

B. Low-Dimensionality Feature-Detection is better

Another interesting conclusion, we can draw from the Pareto points is the fact that most of them, all except the one with the lowest *RF*, use simple features such as mean, minimum, maximum, and variance. These features performed very fast compared to the more complex features such as DCT, FFT, and the wavelet transformations. Even though both the wavelet transformations are O(N), just like the simple features, the wavelets are represented by a vector of coefficients, rather than a scalar, which the simple features use. This is a key difference in the runtime between the two groups.

Fig. 5 shows the Pareto optimal points for each feature and the best overall. The best overall curve, shown in red, is the same as the curve shown in Fig. 4. Notice how the simple feature group and the more complex feature group have similar

TABLE 3: PARETO OPTIMAL POINTS SUMMARY. THIS TABLE SUMMARIZES THE SYSTEM PARAMETERS OF THE HIGHLIGHTED POINTS IN FIGURE 4. NOTE SIMILARITIES IN SIGNAL, FEATURE, FREQUENCY, AND FRAME SIZE. THE TWO COMBINATIONS IN BOLD REPRESENT THE TWO POINTS IN THE "KNEE" OF THE SOLID RED LINES IN FIGURES 4 AND 5.

RF	Norm. Time	Signal (axis)	Feature	Freq. (Hz)	Frame Size	Window Size (s)
0.036	0.2172	x	DCT	100	10	16
0.086	**0.0388**	**y**	**min**	**100**	**20**	**18**
0.112	**0.0359**	**x**	**mean**	**100**	**20**	**16**
0.146	0.0331	y	max	100	20	14
0.170	0.0330	x	min	100	20	14
0.196	0.0216	x	max	100	20	8
0.270	0.0176	x	min	100	20	6
0.340	0.0172	x	max	100	20	6
0.729	0.0059	x	variance	20	20	10
0.754	0.0056	x	variance	20	20	8
0.775	0.0041	x	min	20	20	10
0.829	0.0037	x	mean	20	20	8
0.878	0.0037	z	min	20	20	6
0.882	0.0032	x	mean	20	20	6
0.938	0.0029	x	max	20	20	6

curves within their group. The increase in runtime in the complex feature group is attributable to the increased feature dimension. Equation 9 shows that computational complexity is proportional to the square and cube of the dimension of the feature. These two elements dominate the equation when the dimension of the feature is high. Fig. 6 shows how the number of frames per window and the dimension of the feature affect computational complexity. It shows the graph of $N_f \cdot D^2 + D^3$,

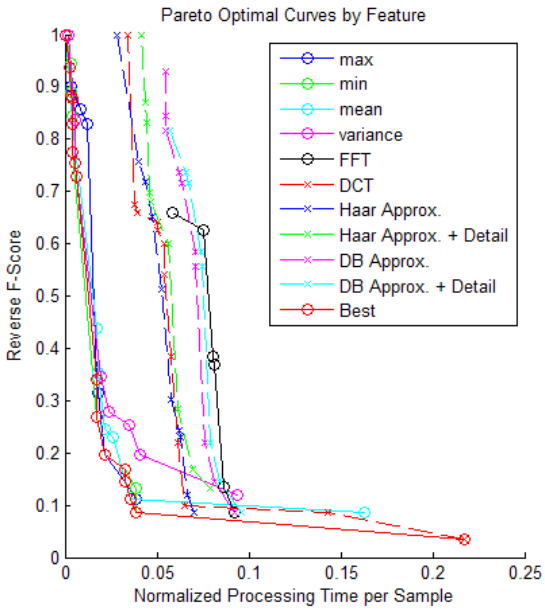

Figure 5. The Pareto optimal points for each feature are shown. This figure shows how feature computational complexity affects system runtime. FFT, DCT, and the wavelet approximations are vectors, but max, min, mean and variance are scalars.

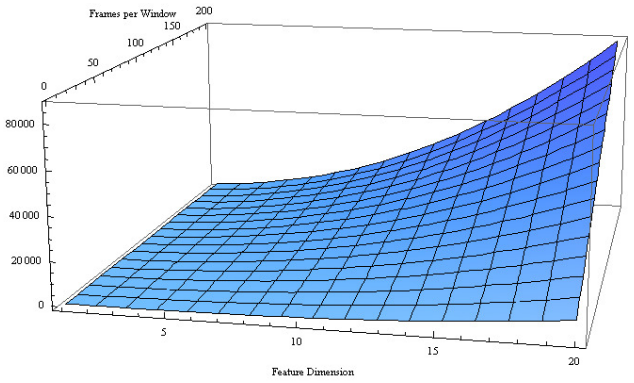

Figure 6. The Relationship between Frames per Window, Feature Dimension and Computational Complexity. Complexity increases sharply with the number of frames per window when feature dimension in high, but is relatively flat when feature dimension is low.

where N_f is the number of frames per window and D is the feature dimension. In our experiments, the feature dimension ranges in size from 1 to 20, and the number of frames per window ranges from 3 to 200. Low feature dimensions have little effect on complexity as the number of frames per window increases. However, a high feature dimension has a significant impact on computational complexity as the number of frames per window increases.

VIII. Conclusion

We have presented a low power approach to detecting activity transitions. The entire framework has several variables and has a corresponding large design space. We have shown that a few combinations of these variables have been effective in detecting transitions in our test cases. We have shown that there is a power and accuracy trade-off when selecting combinations of variables, namely some combinations are very accurate, but they are also computationally complex and use more power. Some combinations sacrifice accuracy only marginally but are much less computationally complex and therefore more power efficient. Previous work in activity recognition has focused on decreasing the sampling frequency to conserve power. Here we have shown that the feature, or more specifically the dimension of the feature, extracted from the signal has a significant impact on the computational complexity and power consumption of the system.

Future work will need to show whether these combinations and this framework for detecting activity transitions scales to activities beyond what we've tested. It may be that the simple features worked well for the small set of activities we tested because their signal signatures were sufficiently distinct from one another. More complex features may be necessary as the number of activities and transitions increases. Also, future work will test whether the same combination of variables that detect transitions can also correctly classify what activity is taking place. The method presented here only detects that a transition occurs, not what the person has transitioned from or to. It may be the case that a low-power activity detection system uses simple features on one signal to detect a transition, but then needs to use a more complex feature on multiple signals to classify the activity.

References

[1] French, B., Siewiorek, D., Smailagic, A., Deisher, M. Selective Sampling Strategies to Conserver Power in Context Aware Devices. In Proceedings of 11th IEEE International Symposium on Wearable Computers (Boston, MA, USA, October 11-13, 2007). ISWC '07. IEEE Computer Society, Los Alamitos, CA, 77-80.

[2] Huynh, T., Fritz, M., and Schiele, B. Discovery of activity patterns using topic models. In Proceedings of UbiComp 2008, ACM Press (2008), 10-19.

[3] Katz, S., Ford, A.B., Moskowitz, R.W., Jackson, B.A., Jaffe, M.W., Studies of Illness in the Aged. The Index of ADL: A Standardized Measure of Biological and Psychosocial Function. Journal of the American Medical Association, 1963 Sep 21, 185:914-919.

[4] Krause, A., Ihmig, M., Rankin, E., Leong, D., Gupta, S., Siewiorek, D., Smailagic, A., Deisher, M., Sengupta U. Trading off Prediction Accuracy and Power Consumption for Context-Aware Wearable Computing. In Proceedings of the Ninth IEEE International Symposium on Wearable Computers (Osaka, Japan, October 18-21, 2005). ISWC '05. IEEE Computer Society, Los Alamitos, CA, 20-26.

[5] Nyan , M.N., Tay, F.E.H., Seah, K.H.W, Sitoh, Y.Y., Classification of gait patterns in the time-frequency domain, Journal of Biomechanics, Volume 39, Issue 14, 2006, Pages 2647-2656.

[6] Sekine, M., Tamura T., Togawa, T., Fukui, Y., Classification of waist-acceleration signals in a continuous walking record, Medical Engineering & Physics, Volume 22, Issue 4, May 2000, Pages 285-291.

[7] Stäger, M., Lukowicz, P., Tröster, G. Power and accuracy trade-offs in sound-based context recognition systems. Pervasive and Mobile Computing 3 (2007), 300-327.

[8] K. Zotos, A. Litke, A. Chatzigeorgiou, S. Nikolaidis, G. Stephanides, Energy Complexity of Software in Embedded Systems, IASTED International Conference on Automation, Control and Applications (ACIT-ACA 2005), Novosibirsk, Russia, June 20-24, 2005.

[9] L. Bao and S. S. Intille, Activity recognition from user-annotated acceleration data, in Proc. PERVASIVE 2004, vol. LNCS 3001, A. Ferscha and F. Mattern, Eds., Berlin, Heidelberg, Germany, 2004, 1–17.

IX. Appendix

This appendix describes the derivation of the ratio test's complexity, seen in Equation 9. Let D be the dimensionality of the feature we are analyzing. Let N_f be the number of frames per window. Calculation of the probability p in Equation 1 involves calculating the mean feature vector ($\boldsymbol{\mu}$) and covariance matrix ($\boldsymbol{\Lambda}$) across the entire window. The complexity of $\boldsymbol{\mu}$ is $O(D \cdot N_f)$, because features are calculated per frame and each dimension of the feature is averaged. The complexity of $\boldsymbol{\Lambda}$ is $O(D^2 \cdot N_f)$. The calculation of p also involves calculating $\boldsymbol{\Lambda}^{-1}$, which has complexity $O(D^3)$ since the best known algorithms to calculate the inverse of a matrix are cubic. The complexity for calculating p can now be simplified to $O(D^3)+O(D^2 \cdot N_f)$.