Don’t Slow Me Down: Bringing energy efficiency to continuous gesture recognition

Giuseppe Raffa\textsuperscript{1}, Jinwon Lee\textsuperscript{2}, Lama Nachman\textsuperscript{1}, Junehwa Song\textsuperscript{2}

\textsuperscript{1} Interaction & Experience Research, Intel Labs, Santa Clara, CA
giuseppe.raffa@intel.com, lama.nachman@intel.com
\textsuperscript{2} Department of Computer Science, Korea Advanced Institute of Science and Technology
jcircle@nclab.kaist.ac.kr, junesong@kaist.ac.kr

Abstract
Gesture is a compelling user interaction modality for enabling truly on-the-go interactions. Unlike keyboard and touch screen interactions which require considerable visual attention and impose stringent constraints on the form factor of mobile devices, people can easily use hand gestures to perform simple actions (e.g. retrieve voice mail) without having to slow down.

In this paper we present an efficient gesture recognition pipeline optimized for “continuous” recognition while minimizing processing overhead and enhancing usability by not requiring the user to delimit explicitly the start and end of gestures. The pipeline is constructed to allow for early filtering of unwanted sensor data with minimal processing cost, and limiting the invocation of processing intensive stages (i.e. HMM) to a limited subset of data (< 5% of sensor data). We also present our evaluation results from a 10 user experiment using 17 gestures and demonstrate that we can achieve considerable processing and power saving without impacting overall recognition accuracy.

1. Introduction
Gestures is a compelling user interaction modality for enabling truly on-the-go interactions. Unlike keyboard and touch screen interactions which require considerable visual attention and impose stringent constraints on the form factor of mobile devices, people can easily use hand gestures to perform simple actions (e.g. retrieve voice mail) without having to slow down. Furthermore, they can receive confirmations or results of these actions using audio feedback from an ear-piece or directly from the phone.

To improve accuracy and reduce computational overhead of gesture recognition, much of the existing work relies on having users clearly indicate the start and stop of gestures, which results in burdening the user. To solve the usability issue while addressing the accuracy and power constraints, we developed an efficient processing pipeline for continuous gesture recognition based on 3D accelerometer and gyroscope sensors. The front-end of the pipeline is comprised of low computation stages responsible for spotting potential gestures and filtering out most of the “non-gestural” sensor data. These stages can be offloaded to a low power processor close to the sensors for further energy optimization. The backend of the pipeline is responsible for recognizing the specific gesture and is more computationally intensive, but will be invoked for only a fraction of the time.

In this paper, we describe our gesture recognition architecture and present extensive experimental results to show the processing and energy advantages of using this architecture without negatively impacting accuracy. We evaluated our gesture recognition pipeline using 17 gestures from 10 users collecting over 5000 gesture samples and over 2000 movements from an 8-hour data collection session. Our analysis demonstrate that with a minimal impact on false negatives in early pipeline stages (0.1%) we achieve the same accuracy on the gesture set as state-of-the-art recognition (99%) with substantial power savings.

The main contributions of this paper are (1) a novel processing pipeline, (2) a novel gesture end point detection improving accuracy and robustness to mobility conditions, and (3) a template matching based method for early rejection of non-gesture movements and late validation of the recognized gesture, (4) analysis and selection of HMM “garbage” model for filtering out non-gestural movements and finally (5) the implementation of a wearable ensemble prototype with several gesture-controlled mobile applications.

In section 2 we review related work on gesture recognition, followed by a description of our approach in section 3. We then describe the pipeline recognition...
stages in section 4 and present an evaluation of the system in section 5. We conclude the paper and discuss future work in section 6.

2. Related Work

For the last two decades, gesture recognition has been extensively investigated [2][3]. A rich body of work focused on vision-based gesture recognition using stationary [17] or wearable cameras [22], and camera-enabled mobile phone [23]. These solutions are not ideal in real mobile environments due to limited wear-ability, high computation and energy requirement, and sensitivity to light conditions [2][3].

Recently, hand-worn accelerometer and gyroscope based gesture recognition have been successfully used in mobile and pervasive computing. These sensors have the advantage of being low cost, low power and extremely compact. Some existing projects have proposed gesture recognition systems based on diverse sensing form factors, e.g., Wii Controller [4], VTT Soapbox [5][6], GeorgiaTech Watch [7], KAIST Ring [8], and ETH Wearable Computer [2]. These systems have different purposes including controlling appliances and writing numbers in the air. Most existing systems do not focus on continuous gesture recognition. Instead, they rely on manual annotation to indicate the start and stop of gesture, e.g., button push/release [4][7]. To alleviate the inconvenience of manual annotations, we designed our system to automatically detect potential gestures from continuous sensor data streams.

In typical systems, sensor devices capture hand motion at a sampling rate of 100 Hz and continuously transmit the raw sensing data to the mobile device, wasting wireless channel bandwidth and energy. In addition, the mobile device continuously invokes the computation-intensive gesture recognition stage based on HMM [2][5][6], SVM [12], or Bayesian Networks [15], resulting in extra processing overhead. To address these limitations, we developed an efficient gesture processing pipeline that filters the majority of non-gestures using low computation stages, and restricts the invocation of computationally-intensive stages to mostly real gestures. Furthermore, we carefully offload the low computation stages to the resource-limited sensor device. As a result, we can reduce the sensor data sent over the wireless channel and enable duty-cycling of the mobile device, hence reducing the overall energy consumption of the system.

Hidden Markov Model (HMM) is commonly used to recognize gestures since it has been shown to have the best accuracy [3][5][6]. However, HMM is computationally intensive due to costly probability calculation. To reduce the computation cost, much simpler algorithms such as DTW [3], Template matching [1][9], and Decision Tree [13][14] have been proposed. However, the recognition accuracy of such simple algorithms is generally much lower than HMM, (e.g., 93.5% in DTW, compared to 98.5% in HMM [3]). In [24], authors developed watch-based recognition system based on discrete HMM. However, the vocabulary is limited to three gestures with low accuracy, i.e., 93.5%. High accuracy is crucial in gesture-based UI, where false negatives will result in unresponsiveness, and false positives will result in undesired actions, both serious UI issues.

Some gesture segmentation methods have been used to automatically detect gestures occurring sporadically in continuous sensor streams. They can be classified into threshold-based [8][10], motion dynamics analysis-based [2], and HMM-based methods [11]. The threshold-based method is light-weight, whereas the other two methods require costly sequence analysis [2] and probability calculation [11], respectively. Our segmentation method is similar to the threshold-based method. However, it achieves higher filtering rate by efficiently avoiding false segmentation, which could be easily generated by non-gestural movement noises.

Power optimization has been studied in sensor-based context-aware systems. In most of systems [25][26], energy efficiency is achieved through efficient selection of sensor sampling rates and features, for optimal energy-accuracy tradeoff. Recently, several methods were proposed to reduce energy without much accuracy degradation, e.g., selective sampling strategies for activity detection [27], and power-efficient decision tree for gait monitoring [28]. While these techniques could be complementary to our approach, our system is specialized in HMM-based gesture recognition and its pipeline architecture optimization.

3. Approach and System Infrastructure

Our gesture recognition prototype (Figure 1) is comprised of two subsystems, a sensor-enabled wrist-worn watch (Context Watch) developed internally, and a commercial off-the-shelf mobile device. The two communicate through a Bluetooth wireless link. We also applied the same pipelined approach to sensors embedded in the device in order to enable gesture interaction by directly moving the device.

![Figure 1. System Prototype](image_url)
3.1. System Prototype

The Context Watch has been developed as a unified mobile sensing platform for diverse context-aware applications. It includes many sensors (e.g. light, audio, compass) of which the 3D accelerometer and gyroscope were used for gesture recognition. The CortexM3 processor is equipped with 64 KB of RAM and 512KB of FLASH and can operate at 8 to 72MHz. It also includes a haptic motor, LEDs and an OLED display allowing for a flexible wearable UI.

The mobile device used in the prototype is an off-the-shelf Mobile Internet Device (MID Viliv S5) that uses an Intel Atom CPU (@ 1.33Ghz) with 1GB of RAM running a full featured Windows XP. An earpiece completes the wearable ensemble for enabling on-the-go UI leveraging gesture, audio and haptics.

3.2. System Design

Most of the previous research focused on coarse movement segmentation and gesture recognition techniques leveraging mainly HMM and other statistical analysis methods. Systems are typically implemented using a centralized approach which results in forwarding all sensor data to the “main” recognition stage, usually computationally intensive, hence invoking it for every movement. Furthermore, if the sensor is wirelessly connected to the mobile device, all the raw sensor data needs to be transmitted over the wireless channel.

To overcome these limitations, we propose a pipelined algorithm where the initial stages of the recognition pipeline require minimal computation but result in major data reduction. This enables considerable savings in energy and processing consumption by allowing the main CPU to be in idle state longer while still capturing and analyzing sensor data in the low power unit. Furthermore, this approach has the added benefit of rejecting non-gesture movements, which tends to be a common problem in HMM-based gesture recognition. The recognition pipeline is composed of the following 8 stages: (1) Sensors capture, (2) low pass filtering, (3) Gesture segmentation, (4) Early Template Matching, (5) Normalization, (6) Feature Extraction, (7) HMM + Garbage model and (8) Late Template Matching.

In the Context Watch, we implemented the early stages (stage 1 to 4) of the gesture recognition pipeline, while the late stages (stage 5 to 8) were implemented on the MID. The HMM recognition stage is based on HTK library [16] with modifications to improve latency and performance (see section 4.7). Accelerometer and gyroscope sensors capture hand motion, and time stamped synchronized raw data is transmitted through the wireless link to the MID only when a “gesture-like” movement is detected by the early stages. Upon reception of the complete gesture segment, the MID performs the final compute-intensive stages of the pipeline, and determines if a meaningful gesture has indeed been performed by the user. In this case, the MID will communicate the performed gesture to any “interested” applications running on the MID. We developed several gesture enabled applications on the MID including voice mail navigation, picture/video viewers through Windows Media Player, MS Powerpoint navigation, voice communication through Skype and many others.

4. Gesture Recognition Pipeline

4.1. Sensors capture

In the sensor capture stage, 3D Accelerometer and 3D gyroscope sensors are sampled at 100Hz in the Context Watch.

4.2. Low-pass IIR filtering

Low pass filtering is needed to eliminate human noise (e.g. hand trembling and user mobility) as well as sensor noise. We use an Exponential Moving Average filter due to its effectiveness in filtering the signal noise while preserving gesture’s signal characteristics residing in the low frequency range. The state equation is shown below, with a computational complexity $O(t)$.

\[
S_t = \alpha * X_t + (1 - \alpha) * S_{t-1}
\]

We empirically chose the attenuation factor $\alpha$ to average the last 20 samples (i.e. 200 msec) balancing latency and attenuation of high frequency components.

4.3. Forward/Backward Movement Detection

Accelerometer-based gesture segmentation is a commonly used technique. It approximates the instantaneous Hand Force (HF) exerted as follows:

\[
HF = \text{Abs}(\sqrt{\text{Accel}_x^2 + \text{Accel}_y^2 + \text{Accel}_z^2} - GRavity)
\]

Thus, the HF is zero if there is no hand motion, making it independent of sensor orientation. Most accelerometer-based segmentation methods utilize a threshold mechanism which suffers from the inherent tension between setting a “weak” versus “strong” HF threshold (HF$_t$). A weak HF$_t$ is susceptible to false positives caused by non-gestural movements. On the other hand, a “strong” HF$_t$ limits the detection to only high-energy gestures and results in losing the initial segment of the signal whose force is weaker than HF$_t$. To address these issues, we developed a
“Forward/Backward” Movement Detection (FBMD) method in which a “strong” HF_{IS} is used initially to trigger the detection and two weak thresholds (HF_{ID}/HF_{IB} for forward/backward end point detection) are used to determine the “real” movement end points. In addition, we set a temporal constraint T_{i} on HF_{EF} and HF_{IB} to avoid the splitting of single gesture into multiple segments. FBMD behavior is shown in Figure 2: when HF_{IS} is exceeded, FBMD enters “during-gesture” state and looks at previously buffered data (i.e. looking backward in time) until the sensor data are below HF_{IB} with duration T_{i}. Similarly, it remains in “during-gesture” state until the data are below HF_{IE} with duration T_{e}. FBMD complexity is linear with respect to the movement duration.

![F/B Movement Detection](image)

**Figure 2.** “Hand Force” during an “ear-touch”

### 4.4. Early Template Matching

When FBMD transitions from “during-gesture” state to “no-movement” state, the identified data segment between HF_{IB} and HF_{IE} is forwarded to the “Early Template Matching” (ETM) stage, which determines if some movement’s characteristics are coarsely compatible with the corresponding characteristics of any of the trained gestures. Template Matching is generally considered not reliable for accurate gesture recognition [1][2][9] due to the overlap of multiple gesture templates. However, for coarse binary classification of gesture/non-gesture, we found this technique highly accurate (see section 5).

Figure 3 shows the distribution of gestures versus non-gestural movements collected during different mobile conditions, indicating clear separation between the two clusters.

![Distribution of gestures and non-gestural movements](image)

**Figure 3.** Distribution of gestures and non-gestural movements

In order to minimize False Negatives (valid gestures discarded) we conservatively calculate a min-max bounding box of each feature across all gestures, utilizing the training data. If any of the signal’s features falls outside the min/max bounding box, the gesture is dropped at this stage. In the current prototype we compare movement duration \( T \) and maximum force exerted by the hand \( MaxForce \) to the ranges of \([T_{MIN}, T_{MAX}]\) and \([F_{MIN}, F_{MAX}]\) calculated over all the samples for all the gestures collected. \( MaxForce \) for each gesture \( G_{j} \) with duration \([t_{1}, t_{2}]\) is

\[
MaxForce_{G_{j}} = \max_{t_{1}}^{t_{2}} (HF_{j}) \quad \forall j \in [1, M]
\]

\[
F_{min} = \min_{i} (MaxForce_{G_{j}}) \quad F_{max} = \max_{i} (MaxForce_{G_{j}})
\]

These features are effective in discarding abnormally short/long or low/high force movements. The computational complexity of this stage is \( O(T) \) for the calculation of all features and \( O(N_{features}) \) for the comparison to each feature bounding box.

### 4.5. Normalization

Using raw sensor data for gesture recognition is not optimal for obtaining high accuracy rates for gestures “in vocabulary” as user-to-user and gesture-to-gesture variability can affect the raw data. To reduce the effects of such variability, we applied Gravitational Effect Estimation and Re-sampling and Re-scaling borrowing from previous research [18][19].

**Gravitational Effect Estimation.** The instantaneous accelerometer reading is:

\[
a_{measured}(t) = a_{movement}(t) + a_{gravitational}(t)
\]

During a movement, both components vary, due to dynamic acceleration caused by the movement and the changing tilt of the device. In a ballistic gesture’s movement the best estimate of the gravitational component \( a_{gravitational}(t) \) is the mean of the measured acceleration sequence on each axis [18].

**Re-sampling and Re-scaling.** To alleviate user-to-user and sample-to-sample variability, it is necessary to normalize the signals to compensate for variability during the movement. In the case of signal resampling, we used the maximum gesture length from the training data as output sequence length, to avoid down sampling the signal. Similarly, we performed re-scaling of the signal to the global maximum and minimum values found in the training data set.

### 4.6. Feature Extraction

In addition to raw data, we found first-derivative and integral of the signal to be effective in improving recognition accuracy. This is due to their ability to describe the main characteristics of the signal such as its absolute trending (raw data), its relative change
(delta) and the cumulative effect (integral). Hence, we used the following feature vector FV(t) calculated over the movement segment [t_1,t_2]:

\[
FV (t) = \sum_{i=1}^{N} s(t_i) - s(t_{i-1}) \sum_{i=1}^{N} s(t_i) \sum_{t \in [t_1,t_2]}
\]

s(t_i) is the signal vector constituted by the filtered and normalized 3D accelerometer and gyroscope. Therefore FV contains 18 elements. We select these features as used commonly in other works [5][7].

4.7. HMM + Garbage model (HMM+G)

The core of the recognition algorithm is an HMM-based recognizer [5][20]. Training is performed utilizing standard Baum-Welch EM procedure on the set of gesture’s sample collected. We trained a model for each gesture using samples from all users and a garbage model using all samples. The garbage model concept has been successfully utilized in the past in speech recognition for filtering out-of-vocabulary words utilizing “garbage” speech data and in gesture recognition using the complete training data [11].

During the recognition phase, we apply the Viterbi recognition algorithm to the incoming feature vector FV. The gesture vocabulary is composed of a model for each trained gesture in addition to the garbage model. A “Viterbi score” is computed for each model. The previous rescaling procedure allows for already normalized scores. The HMM stage outputs the best candidate as well as the garbage model score. A model M is assumed to be the recognized gesture if its score exceeds the score of the garbage model. In our system we use left-right HMM topology and continuous output density probability functions modeled as Gaussian mixtures, leading to better performance than discrete output probabilities. We performed several experiments varying the number of states and the number of mixtures and found that 8 states, 0 skip states, and 3 Gaussian mixtures to be optimal for achieving highest accuracy.

4.8. Late Template Matching (LTM)

Template Matching is also applied to the original signal of the predicted gesture to validate the HMM result and further reduce false positives with a minimal impact on false negatives. This stage uses the same min/max bounding box method as ETM; however it will match the recognized gesture’s features against the stored Template of the candidate gesture only. If the candidate gesture’s features match the Template, the final gesture event is triggered. In Figure 4 Duration and MaxForce distributions for two gestures are shown. This approach is effective because the HMM Viterbi score is effective only as a relative measure (as opposed to absolute likelihood) [21].

5. System Evaluation

In this section, we present an evaluation of our system and highlight the effectiveness of each of the pipeline stages in filtering unwanted data. We use a high-fidelity HMM as a baseline, and analyze the impact of our pipeline approach on the overall accuracy. We demonstrate that the proposed system achieves a comparable level of accuracy while greatly increasing the overall energy and computation efficiency. Finally we quantify bandwidth and computation efficiency.

5.1. Data collection methodology

Compelling on-the-go user interfaces require careful selection of gestures. In our evaluation, we chose 17 gestures (Figure 5). Some gestures (e.g. circle) are characterized by long duration and low power exerted by the user while others (e.g. left rotation) show the opposite. Using such a diverse set of gestures ensures broader applicability of our system to new gestures.

The evaluation is based on data collection of the 17 gestures from 10 users (5 women and 5 men with varying height, weight and physical characteristics) over two sessions in different days to avoid overfitting. We collected over 5,000 gesture samples (30 samples per gesture per user) in standing conditions and different postures as well as slow movements to increase variability in the data set. The users were asked to comfortably wear the watch and freely
perform the gestures after seeing one example. We also collected movement data in 6 different settings (see Table 2), for a total of ~2000 movement segments in 8 hours of data collection. In these settings, gestures in the data set were carefully avoided, while involuntary movements as well as movements due to the particular activity were performed. This data can be utilized to evaluate the efficiency of various stages of the pipeline in filtering out “non gestural” movements.

5.2. Pipeline performance

In order to quantify the performance of the pipeline, we exercised the pipeline using the collected gestures and movements. In Table 1, we show the rate of data discarded by each stage in both situations (gestures data on the right and movement data on the left) as well as the pipeline accuracy in correctly recognizing gestures. In this paper we use the following measures:

- **Discard Rate (DR)** and False Positive Rate (FP) refer respectively to the duration of movements discarded and duration of movements erroneously not discarded over the total duration of movements coming in as input in each stage. For “non gesture” movements holds FP=1-DR
- **False Negative Rate (FN)** and Accuracy (ACC) refer respectively to #gestures erroneously discarded and #gestures correctly recognized over the total number of gestures collected.

**Table 1. Pipeline performance evaluation**

<table>
<thead>
<tr>
<th>Pipeline Performance Evaluation</th>
<th>DR%</th>
<th>FP%</th>
<th>FN%</th>
<th>ACC%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FBMD</td>
<td>72</td>
<td>28</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>ETM</td>
<td>96</td>
<td>4</td>
<td>0.1</td>
<td>N/A</td>
</tr>
<tr>
<td>HMM+G</td>
<td>88</td>
<td>12</td>
<td>0.5</td>
<td>98.8</td>
</tr>
<tr>
<td>LTM</td>
<td>98</td>
<td>2</td>
<td>0.1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

FBMD discards on average more than 70% of movements, whilst not losing any gesture. Of the remaining data, ETM filters out most of the remaining movements (96%) while incurring a 0.1% false negatives (discarded gestures). Therefore we achieve a 98% saving of non-gesture movement not passed to the MID for further processing while incurring only a 0.1% false negatives. This data reduction results in improving communication and processing efficiency by reducing the invocation of high computation stages and opportunistically duty cycling the MID.

HMM+G achieves 98.8% accuracy on spotted gestures at a cost of 0.5% rejection rate of valid gestures by the garbage model. The remaining 0.7% of spotted gestures are incorrectly classified by the HMM.

As for the rejection of movements passed through the pipeline’s early stages, HMM+G garbage model is effective in filtering 88% of them. Finally LTM filters out most of the remaining movements (98%) at a cost of 0.1% FN to real gestures. Hence, the overall error introduced by the pipeline is 0.2% False Negatives (0.1% for ETM + 0.1% for LTM). However, the pipeline is able to filter 98% of the movements.

In the following sections we highlight experiments that we conducted in order to optimize specific pipeline stages and ensure robustness in various mobility conditions.

5.3. Movement Detection performance

The main advantage of our movement detection over a naïve threshold-based approach is the improved robustness with respect to non-gestural movements. Performance has been evaluated by measuring the discard rate of non-gestural movements. We empirically determined the value of the temporal threshold T_t to be 200 ms, hence capturing the entire gesture segment without splitting it into multiple movement segments. As for the initial threshold, we set k=0.8 to allow for variability in the MaxForce of the gesture performed.

**Table 2. FBMD discard rate**

<table>
<thead>
<tr>
<th>FBMD Evaluation</th>
<th>Situation</th>
<th>Discard rate [%]</th>
<th>Situation</th>
<th>Discard rate [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FBMD</td>
<td>Driving</td>
<td>83.2</td>
<td>Plane</td>
<td>76.5</td>
</tr>
<tr>
<td></td>
<td>Riding Bus</td>
<td>87.4</td>
<td>Computer work</td>
<td>84.9</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>72.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows that on average 72% of all non-gesture movements are discarded by this stage. We also assessed the performance under several mobility conditions and discovered that even during high mobility, this stage discards more than 50% of the data. We also confirmed that a high HF_k (k=0.8) does not trigger false negatives (only 1 gesture was discarded on a 10-fold analysis). On the contrary, with a single-threshold approach, a low HF_k (i.e. k=0.1) is needed in order to capture 90% excursion of the signal for correct processing. In this case, the average DR rate has been a modest 7% of the movement data, leaving 93% of the movement data passing through.

5.4. Template Matching Performance

To evaluate ETM, we explored the following 3 low computation algorithms:
• ETM1: using a bounding box based on the min-max values of the two features, calculated over the training data.
• ETM2: using a bounding box based on extremes 
  
  \[ \text{Average(feature)} \pm 3 \times \text{StandardDeviation(feature)} \]
• ETM3: using a C4.5 Decision Tree.

Results from 10-fold evaluation (Table 3) show that all methods offer excellent classification of gestures vs. non-gestural movements, even under non stationary conditions such as walking and driving. While ETM3 offers the best accuracy, we chose ETM1 because it results in almost 0% FN with only a small increase of 3.20% in FP with respect to ETM2. Since further filtering of FP will be performed at later stages, it is more important to minimize FN at this stage.

<table>
<thead>
<tr>
<th>ETM Evaluation</th>
<th>FP(%)</th>
<th>FN(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETM1 - Min-Max</td>
<td>3.62</td>
<td>0.1</td>
</tr>
<tr>
<td>ETM2 - Avg±3StdDev</td>
<td>1.44</td>
<td>1.93</td>
</tr>
<tr>
<td>ETM3 - DecisionTree</td>
<td>2.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Our results also show that FP increase as the physical activity increases in intensity; for example FP in the office are close to 3%, and increase to about 5.5% when commuting or walking.

5.5. HMM+G accuracy

We tested the accuracy of the HMM subsystem in terms of correctly recognized instances over the entire gestures’ sample set used for testing the model without utilizing any garbage data. We used 10-fold cross validation over the 5000 gesture samples, not accounting for non-gestural movements. We found that the gyroscope can improve the accuracy on average 2%, with a maximum of 4% for the gesture “circle”. In addition, we discovered that limiting the training to data from one day while testing with data from another day lowers the accuracy of the system by 2%, due to over-fitting and decreased variability in the gesture samples. This indicates the importance of variability in the training data and a possible need for continuous learning which we plan to explore in the future. The average accuracy of the data set is 98.8% and the range of accuracies goes from 97.4% for “Right Rotation” to 100% for “EarTouch”. We tested user-independent accuracy performing a “leave one (user) out” analysis, which reported similar average accuracy (97.7%). We did not find significant differences using more than 8 states. The analysis of the confusion matrix highlighted that the garbage model is the main source of misrecognition, filtering out ~0.5% of legitimate gestures (Table 4) for the chosen GM1.

To determine the best approach for the construction of the garbage model, we analyzed 3 different strategies (GM1-3), all of which use the same HMM topology as the gestures’ models. Table 4 compares their performance in terms of False Positives and False Negatives. In this experiment we did not utilize other pipeline stages in order to isolate the effectiveness of the Garbage model. We observed that garbage model GM2 built on top of real “garbage” data (i.e. non-gesture movements) reduces False Positives relative to baseline GM1 which uses all samples. However, GM2 reduces the accuracy of the HMM when recognizing real gestures (FN increases). Similarly, GM3 which uses a mix of gestures and non-gesture data is somewhat between GM1 and GM3 for both FP and FN. We decided to use GM1 for two reasons: (A) It minimizes false negatives and (B) since it has been built using the trained gestures, it naturally acts as a “threshold” against movements that are “similar” to the trained gestures, which would not be filtered by LTM, making it complimentary to Late Template Matching.

<table>
<thead>
<tr>
<th>Garbage Model Evaluation</th>
<th>FP%</th>
<th>FN%</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM1: All Sample data</td>
<td>12.3</td>
<td>0.5</td>
</tr>
<tr>
<td>GM2: Garbage data</td>
<td>7</td>
<td>1.8</td>
</tr>
<tr>
<td>GM3: Sample + Garbage data</td>
<td>9.5</td>
<td>1.6</td>
</tr>
</tbody>
</table>

5.6. Bandwidth and processing savings

We implemented FBMD and ETM in the Context Watch and verified that these algorithms can run in real time at the lowest frequency allowed by the processor (8MHz). Power consumption of the watch is limited to 80 mW during this mode. When data is being transferred over Bluetooth to the MID (i.e. when a possible gesture is detected by the watch), power consumption rises to 180 mW. On the other hand, we measured the power consumption of the MID to be on order of 6W while fully processing HMM stages and estimated about 2 W in idle mode. As a result, keeping the MID in idle mode will reduce the power consumption of the system by about 4 W when the sensor data is being filtered by the watch. Cell phones have a lower power consumption profile than MIDs, but are much more constrained in computation capabilities, making HMM a very challenging workload. We haven’t implemented our workload on the cell phone, so we can’t state feasibility or power consumption numbers. However, we still expect substantial power savings even on such lower power platforms. For example, our back of the envelop calculations using the HTC touch Pro phone shows about 400 mW of power consumption in standby...
mode, and about 1.2 W while actively processing sensor data, resulting in about 700 mW of power savings when data is being filtered by the watch.

6. Conclusion and Future Work

Gesture recognition has great potential in enabling on-the-go interactions as it has an advantage over traditional interfaces of not requiring visual attention. However such potential comes at a computation, power consumption and accuracy costs. Traditional approaches relied on explicit user indication of starting a gesture to reduce such costs, however, we believe these solutions have negative impact on usability, and we sought to solve the problem differently.

In this paper we presented a novel approach to gesture recognition, which exploits low-computation algorithms in order to improve system performance especially in battery constrained devices and alleviate recognition errors that often frustrate users. We demonstrated that such a system can be efficiently implemented in a pipeline where the initial low-computation stages are offloaded to the sensor node. We proposed novel approaches in movement detection, template matching stages, and analyzed different methodologies to improve HMM garbage model construction. We collected gestures data from 10 users for a total of 5000 gestures and 8 hours of sensors data while not performing any gesture. Experimental results showed the effectiveness of this approach, achieving the same accuracy on recognizing valid gestures while substantially decreasing the data transmission and computation, hence enhancing battery life and optimizing computing resources.

In the future, we plan to explore the role of context-awareness in improving gesture recognition (e.g. physical activity and social situation.) From a usability point of view, a review of the gesture vocabulary is warranted, in order to understand user needs and preferences under various conditions.

References