Exploration of Interactions Detectable by Wearable IMU Sensors

Rajesh Kuni, Yashaswini Prathivadi, Jian Wu, Terrell R. Bennett, Roozbeh Jafari
Department of Electrical Engineering
University of Texas at Dallas
Richardson, USA
{rxk103020, yxp120330, jian.wu, tbennett, rjafari}@utdallas.edu

***Abstract*—Context aware systems like smart homes and offices will benefit from determining human-object and human-human interactions. In this paper, we explore interaction detection methods using only wearable Inertial Measurement Units (IMUs). The interactions we explore involve two actors - the primary person and a secondary object or person. We explore how several commonly used time domain signal processing operators can be utilized to detect the similar movements in the interactions and thus the interactions themselves. We also utilize a well-known boosting algorithm to potentially increase the accuracy of the operator results. The techniques operate on the magnitudes of the acceleration and gyroscope readings to keep the analysis independent of the orientation of the sensors. The detection accuracy for six interactions using the approach presented in the paper range from 84.2% to 69.6%.**

Keywords—Human-object and human-human interactions, IMU wearable sensors, similar movement, AdaBoost algorithm

I. INTRODUCTION

Context aware computing has the potential to improve the everyday lives of humans through effective resource management, providing intuitive user interface and enabling seamless communication among devices that are aware of the context [1]. Therefore, it has been the focus of research specifically with the growing presence of IoT and wearable computers. A context aware system offers services that can adapt to a changing environment [2]. Context aware systems have many applications in the healthcare segment where they are used to improve the well-being of people by assisting them in their day to day lives [3, 4]. Importantly, a context aware system offers services that detect and mitigate a problem before it worsens because it can detect irregularities in an environment [5]. Context is information that can be used to characterize the current situation of the surrounding environment [1, 5]. The goal of a context aware system is to acquire the context information describing the world around it and make a decision that will benefit the people it serves [6].

Humans go through countless interactions with objects and people that surround their daily lives. For example, on any given day a person picks up his phone, shakes a colleague's hand, opens up his laptop, carries his bag, etc. Because these daily interactions happen so often, they make up a large part of a person's overall context. Therefore, accurately detecting daily human interactions is essential for a context aware system. In this paper we explore methods to detect interactions leveraging IMU-based wearable computers. We consider an actor to be either a person or an object with IMU sensor attached. The intuition is to identify similar movements on IMU sensors worn by the actors. The IMU sensor consists of a 3-axis accelerometer and 3-axis gyroscope in a compact package that affords ease of use, is power efficient, and is small enough to be worn on the body [7]. IMUs are also likely to be more accepted by the public because they can be turned on or off or removed at any time by the user. [8]. All these traits make IMU sensors well suited for the purpose of this paper-*to explore the detection of daily human interactions* .In order to perform this exploration, we first extract features of interest from the magnitudes and magnitude derivatives of the raw IMU data by using the operators described in Section IV. Then we check the operator values against different thresholds to generate numerous weak binary classifiers. These results are provided to the AdaBoost algorithm (described in Section V) to generate a strong binary classifier which is then compared against the ground truth to assess the accuracy of the detection [9].

The main limitation of the techniques explored in this paper comes from the fact that we use time domain signal processing operators. If the IMUs are not synchronized and their clocks drift apart with time, the results of the operators will be less accurate because the timing of the samples is inaccurate. Therefore, we cannot expect the operators to work properly if the sensors are not synchronized. Another limitation is the sensor power source. Despite their power efficiency, the sensor batteries will have to inevitably be replaced. Periodically recharging sensors placed on objects is infeasible. Some solutions to extend the battery life as much as possible include transmitting data intermittently or switching the sensors to low power mode when no activity is detected for a period of time. A few new solutions are offering μW power budgets for motion sensors[1]. Therefore, foreseeing devices with batteries that would last years is not unreasonable. However, the purpose of this paper is not to explore solutions to the inherent weakness of portable power sources; but instead to explore methods that would determine interactions between two actors.

The body of the paper is organized as follows. In Section II, we discuss related works. In Section III, we introduce the types of interactions of interest in this investigation. In Section IV we explain how the signal processing operators

[1] http://www.analog.com/library/analogdialogue/archives/48-12/wearable_electronics.html

are used to generate the weak classifiers. Section V describes how the AdaBoost algorithm is applied. In Section VI, we discuss experimental procedures and results. We conclude in Section VII and briefly discuss future work.

II. Related Works

A subset of context aware computing research has focused on recognizing human interactions and activities. Some researchers have explored activity and interaction recognitions using just video sensors. This approach leads to major problems in an uncontrolled environment because the quality of visual information can drastically vary due to factors like viewing angle, lighting, and the way interactions are performed [10, 11]. Other investigations attempt to alleviate this problem by combining visual sensors with audio sensors to get more accurate readings of the environment [10]. However, users may feel uncomfortable having cameras and microphones recording their daily activities [12]. There have also been works that leveraged Microsoft Kinect videos from which human 3D poses can be detected more accurately with additional depth information [13]. However, this system still requires an elaborate setup that is infeasible in situations where monitoring must be done in a continuous manner. Our work takes a different approach by using IMUs for interaction recognition. The IMU is more portable than a Kinect, allowing for continuous monitoring. It also does not require a complex setup - the sensors are simply attached to a subject and the data is streamed to a laptop for later processing via Bluetooth, Bluetooth low energy or Zigbee. The data can also be stored on wearable devices with reasonably accurate time stamps and retrieved later to processing. In other works, a person's interaction with the environment is modeled as a simple yes/no based interaction. For example, object interactions are recognized using simple sensors placed all around the home to detect simple state changes [12]. Analyzing the state changes from a set of these sensors would unveil the person's routine for that day. Investigations have analyzed how simple if-this-then-that trigger programming has been used to implement smart home systems by users of the smart homes themselves [14]. Our work is not focused on creating a high level picture of person's daily routine but instead on accurately detecting fine-grained interactions. Some studies have used IMUs in smart phones to detect human activities such as standing, walking, lying down, and running [15]. This differs from our work because instead of focusing on individual activity recognition, we look at interactions that involve two actors. There have been works that look at interactions between two actors. Yet, both the actors are human and the work uses vision sensors [13].

Our work is unique from the mentioned studies in several ways. We do not use cameras or microphones to detect human-human interactions. We instead only rely on IMUs. We differ from works that use simple binary sensors to detect interactions because our work can answer the question of who is interacting with what. To the best of our knowledge, this is the first investigation aimed at detecting human-object interactions at a finer level using IMUs.

III. Interactions

There are two categories of interactions we focus on. The first category is any interaction between a person and an object. Examples of such interactions include a person picking up a cell phone, carrying a backpack, pushing and pulling a chair, and writing with a pen. The second category is any interaction between two individuals. Examples of such interactions include shaking hands with colleagues, waving to each other, and walking together. Each of these interactions involve two actors - the primary and secondary actor. The primary actor is defined as the human in the human-object interaction and the secondary actor is defined as the object. As for the human-human interaction, any subject can be defined as a primary actor and the other one will be defined as secondary actor.

The actors involved in the interaction experience similar movements. For example, when a person picks up his phone, the sensor on the phone and the sensor on the user's hand record similar movements. The steps taken in our approach detect the similar movements to ultimately detect the interaction. Each actor may have multiple sensors attached. Our solution involves comparing all combinations of two sensors such that one sensor is from the primary actor and the other from the secondary actor and then finding out which sensor on the secondary actor best matches the sensors on the primary actor. For example, if we want to detect shaking hands, then it could be that all the sensors on the primary actor best match with a sensor attached to the secondary actor's arm.

A majority of the human-human interactions mentioned above are interactions where no contact is made between the two actors. Despite the lack of contact, the two actors are still going through the same motions. For example, it has been shown that two people walking together exhibit synchronicity in their movements [16]. We believe this synchronized movement can be exploited to detect non-contact interactions.

The human-object interactions we investigate are *picking up a cup*, *carrying a backpack*, *writing with a pen*, and *pushing-pulling a chair*. The human-human interactions we investigate are *shaking hands* and *walking together*.

IV. Signal Processing Operators & Weak Classifiers

The main goal of our analysis is to explore how some common time domain signal processing operators can be used to detect daily interactions. The operators we consider are *Peak Skew*, *Zero Crossing Skew*, *Peak Count Difference*, *Zero Crossing Count Difference*, and *Covariance*. The first part of our approach is described below.

A. Preprocessing

As the very first step, the 3-axis accelerometer and gyroscope sensor readings are preprocessed. The accelerometer and gyroscope readings are filtered using a 3rd order low-pass Butterworth filter with a cut-off frequency of 3 Hz to remove noise. The following data types (DT) are extracted from the readings

$$DT = \{\boldsymbol{AM}, \boldsymbol{GM}, \boldsymbol{AD}, \boldsymbol{GD}\} \quad (1)$$

where $\boldsymbol{AM}$ is accelerometer magnitude, $\boldsymbol{GM}$ is gyroscope magnitude, $\boldsymbol{AD}$ is the derivative of $\boldsymbol{AM}$, and $\boldsymbol{GD}$ is the derivative of $\boldsymbol{GM}$. From each sensor stream $\boldsymbol{AM}$ and $\boldsymbol{GM}$ are calculated as

$$\boldsymbol{AM}\,(i) = \sqrt{\boldsymbol{a_x}(i)^2 + \boldsymbol{a_y}(i)^2 + \boldsymbol{a_z}(i)^2} \quad (2)$$

$$\boldsymbol{GM}\,(i) = \sqrt{\boldsymbol{g_x}(i)^2 + \boldsymbol{g_y}(i)^2 + \boldsymbol{g_z}(i)^2} \quad (3)$$

Where $a_x(i)$, $a_y(i)$ and $a_z(i)$ are the *ith* acceleration readings along *X-axis*, *Y-axis* and *Z-axis*, respectively. $g_x(i)$, $g_y(i)$ and $g_z(i)$ are the gyroscope readings along *X-axis*, *Y-axis* and *Z-axis*, respectively. $\boldsymbol{AD}$ and $\boldsymbol{GD}$ are calculated as

$$\boldsymbol{AD}(i) = \boldsymbol{AM}(i) - \boldsymbol{AM}(i-1) \quad (4)$$

$$\boldsymbol{GD}(i) = \boldsymbol{GM}(i) - \boldsymbol{GM}(i-1) \quad (5)$$

In equations (2) – (5), i is the current sample number. We generate four preprocessed vectors for each sensor involved in an interaction. These vectors are applied to the operators.

B. Operator Description

Every operator takes two time series vectors as input. The vectors can be of any type in (1) as long as both the vectors are of the same type (*e.g.*, both $\boldsymbol{AM}$). Each vector comes from a unique actor during an interaction. The operators are discussed below:

1) Peak Skew and Peak Count Difference

The *Peak Skew* operator quantifies how well peaks from two different vectors are synchronized. Given two finite vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, the operator first finds the indices of the peaks in $\boldsymbol{X}$ and $\boldsymbol{Y}$. It then selects the vector that has more peaks and matches all its peaks to peak indices in the other vector. For example, if $\boldsymbol{X}$ has more peaks than $\boldsymbol{Y}$, then each peak index in $\boldsymbol{X}$ is matched with a peak index in $\boldsymbol{Y}$ such that the absolute time difference between the two index values is minimal. The skew value is the sum of all time differences. The *Peak Count Difference* operator calculates the difference in the number of peaks between the two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$.

2) Zero Crossing Skew, Zero Crossing Count Difference

Before applying these operators, we normalize the signal by subtracting its mean value from all samples. The *Zero-Crossing-Skew* operator measures the synchronicity between the positive and negative zero crossings in two different vectors. The positive crossing is defined as the data point at which the signal goes from negative to positive. The negative crossing is defined as the data point at which the signal goes from positive to negative. Given vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, the operator first finds the indices of the positive crossings in $\boldsymbol{X}$ and $\boldsymbol{Y}$. The indices represent the time instance of a crossing. It selects the vector that has more positive crossings and matches them to the positive crossings in the other vector. If $\boldsymbol{X}$ has more positive crossings than $\boldsymbol{Y}$, then each positive crossing index in $\boldsymbol{X}$ is matched with a positive crossing index in $\boldsymbol{Y}$ such that the absolute time difference between the two index values is minimal. The skew value is the summation of all the differences. To calculate the skew value for negative crossings between $\boldsymbol{X}$ and $\boldsymbol{Y}$, the same procedure mentioned above is applied. The *Zero Crossing Count Difference* operator computes the difference in the number of zero crossings between two vectors (*e.g.* $\boldsymbol{X}$ and $\boldsymbol{Y}$).

3) Covariance

Given two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, the *Covariance* operator quantifies how well the two vector change together.

C. Operator Result Vector

Given two streams $\boldsymbol{S1}$ and $\boldsymbol{S2}$ that are of the same type (DT), at any given time, the operators look at two data windows of equal size from $\boldsymbol{S1}$ and $\boldsymbol{S2}$ denoted as $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. The operator is applied on these two data windows and the result is stored in a separate vector. The windows are then advanced by one data sample (overlapping moving windows). The operator repeats its calculation for the shifted windows of data and stores the result in the next index in the result vector. This procedure is repeated until the windows slide to include the last sample in $\boldsymbol{S1}$ and $\boldsymbol{S2}$.

Fig. 1 shows the graphs for the $\boldsymbol{AM}$ values for *picking up a cup* interaction. The blue solid and green dotted stream represent vectors of data from two sensors. An operator looks at data windows (represented by the red rectangle) from the dotted green and solid blue streams and extracts a feature. This operator result is stored in a new vector. This operator result is a representative of the data window. The red rectangle on the left shows the first data window and the rectangle on the right shows the last data window. Fig. 2 shows the result vector for the covariance operator once the window has slid to the last data sample on dotted green and solid blue streams.

The results of *Peak Skew*, *Peak Count Difference*, *Zero Crossing Skew*, *Zero Crossing Count Difference*, and *Covariance* are stored in vectors $\boldsymbol{PC}$, $\boldsymbol{PCD}$, $\boldsymbol{ZCS}$, $\boldsymbol{ZCD}$, and $\boldsymbol{Cov}$ vectors, respectively.

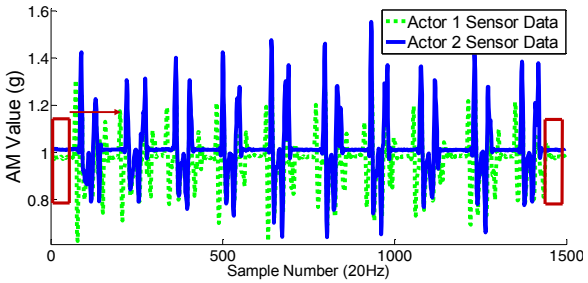

Fig. 1. Data of type $\boldsymbol{AM}$ from two different sensors for *picking up a cup*

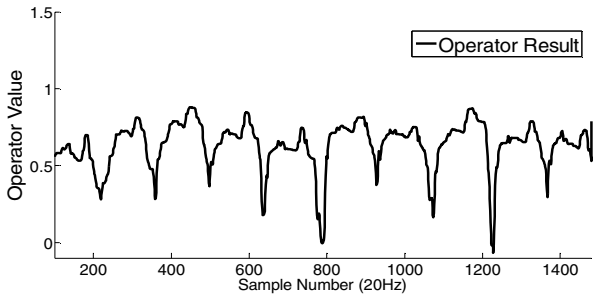

Fig. 2. The black line is the covariance operator result once the operator has evaluated all windows of data

D. Weak Classifiers

The operator results that are extracted from data windows are evaluated against different thresholds. Each threshold acts as a weak classifier, classifying the data window as containing a non-interaction (indicated as -1) or an interaction (indicated as +1) based on whether the operator result that corresponds to that data window is greater or less than the threshold value. Fig. 3 shows the covariance operator result from *picking up a cup* interaction. In this interaction, a subject starts by moving his hand toward the cup, picks it up, and takes a sip from it. Then the subject puts the cup back down and rests his hand on the side of his body.

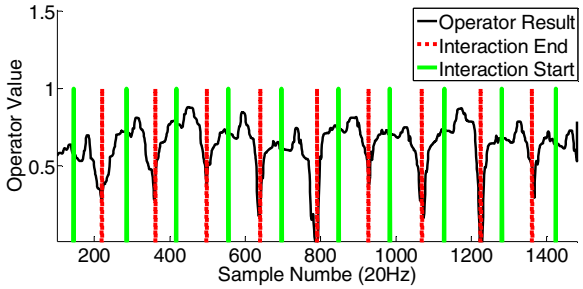

Fig. 3. Covariance results with annotations for *picking up a cup* interaction

The solid vertical green line indicates when an interaction started and the dotted vertical red line indicates when interaction ended. To generate the thresholds, we find the minimum value and maximum value of the operator result in these sections. These two values give a range and this range is divided into ten equal steps. Each of these steps is taken as a threshold value. For ***Cov***, the class of a data window is evaluated to -1 if the corresponding covariance value is less than a threshold value and +1 if it is above a threshold. This is in line with the intuition that if two sensors are experiencing similar movement, their signals will have a higher covariance. For the four remaining operator results, the class is evaluated to +1 if the corresponding operator value is less than a threshold value and -1 if not. This is in line with the intuition that if two sensors experience similar movements, their signals will have a similar number of peaks (peak count difference will be low) occurring at nearly the same time (peak skew value will be low). The same reasoning can be applied to zero cross skew and zero cross count.

Because each of ***PS***, ***PCD***, ***ZCS***, ***ZCD***, and ***Cov*** will be evaluated against ten thresholds, 50 weak classifier vectors will be generated for each data type in DT. Because we are extracting features from four different data types (***AM***, ***GM***, ***AD***, ***GD***), 200 weak classifier vectors will be generated for each sensor pair in an interaction. The total number of weak classifiers that are finally generated for an interaction will depend on the number of sensor pairings between the primary and secondary actors in an interaction.

Once all the weak classifiers are generated, their accuracies are measured by comparing their classification results to the ground truth. Let ***GT*** represent the ground truth vector for an interaction. It is generated from the annotations (illustrated in Fig. 3) and tells us exactly when an interaction occurred. It has the same domain as a weak classifier, $\{-1,1\}$. Let ***Wk*** represent a weak classifier result vector. The accuracy *acc*, of the weak classifier is calculated as

$$acc = \sum_{i=1}^{n} I(\boldsymbol{GT}(i), \boldsymbol{Wk}(i))/n \quad (6)$$

where n represents the number of samples to be classified and I is an indicator function which equals 1 when $\boldsymbol{GT}(i) \neq \boldsymbol{Wk}(i)$ and 0 otherwise. The accuracy of the strong classifier is calculated in the same manner.

V. Boosting Algorithm

Taken by themselves, the weak classifiers may not classify the interaction with a high accuracy. In order to improve the accuracy using these weak classifiers, we use AdaBoost, an algorithm that creates a strong classifier by weighing the decisions of all weak classifiers and then accumulating all the results. Given the generated weak classifiers, we choose only those weak classifiers with accuracy greater than 50% to train the AdaBoost classifier. In other words, a weak classifier must be better than random guessing for a two-class problem [9]. To use the AdaBoost algorithm, we first arrange the results of all the selected weak classifiers in the form of a *result matrix*. Each column in the matrix represents the results of one of the weak classifiers for all the input data windows and each row represents the results of every weak classifier for one input data window. The ground truth vector ***GT*** is also used in the implementation of the algorithm.

In the training phase, AdaBoost starts by giving each data window (*i.e.*, each row) selected for training, equal normalized weights. It then finds the weak classifier which gives the lowest error rate by comparing it to ***GT.*** This error rate is calculated by adding up the weights of those data windows that were misclassified by the weak classifier. AdaBoost gives this weak classifier a classifier weight based on its error rate. All the data windows are then reweighed again based on how the selected weak classifier classified them. If a data window was wrongly classified, it is given more importance (more weight) and if it was correctly classified it is given less importance (less weight). The algorithm then tries to find the next weak classifier with the lowest error rate. This procedure is repeated until all the weak classifiers are given a weight.

In the testing phase, the test data windows from the sensors in an interaction are applied to the weak classifiers and the weak classification results are generated. For each data window, the weak classifier results are weighed by the weights determined by the AdaBoost algorithm. These weighted results are then accumulated. If the result of the accumulation is positive, then the data window is classified as containing an interaction. If negative, the data window is classified as containing a non-interaction. A formal description of AdaBoost is explained by Schapire [9].

VI. Experimental Procedure and Results

A. Sensor Preliminares

The sensor node used for the experiments is a custom built board that consists of a Texas Instruments MSP430 as the microcontroller interfaced to an Invensense MPU-9150, a 9-DOF IMU consisting of a 3-axis accelerometer, 3-axis gyroscope, and a 3-axis magnetometer [17]. We do not use

magnetometers as it is challenging to calibrate them. The IMUs were configured to sample at 20Hz. The literature shows this sampling frequency is sufficient for capturing human movements [18]. Each interaction has sensors strapped to the wrist, arm (above elbow), and ankle. All sensors are placed on dominant side (*i.e.*, right or left-handed) of the subject. Fig. 4a shows the IMU sensor used for the experiments. Fig. 4b shows the placement of the sensors on the body. Fig. 5 shows the placement of the sensors on the different objects.

a. b.

Fig. 4. Left: IMU Sensor Used, Right: Body sensor placements

Fig. 5. Object sensor placements from left to right: sensor on backpack, sensor on chair, sensor in cup, sensor attached to pen

B. Data Collection

The human-object interactions we investigate are *picking up a cup*, *carrying a backpack*, *writing with a pen*, and *pushing-pulling a chair*. The human-human interactions we investigate are *shaking hands* and *walking together*. The human-object interactions are performed by five different subjects who repeat each interaction ten times. The human-human interactions are performed by five unique pairs of subjects. The *shaking hands* interaction is repeated ten times and the *walking together* interaction has five trials of two subjects walking together for approximately 15 seconds.

For training purposes, once the data is collected for each of the above interactions, they are annotated to distinguish non-interaction periods from interaction periods using the data visualization tool introduced in *MoST* [19]. With this tool, the beginning and the end of an interaction are determined manually using the video that is captured during the data collection. This will establish the ground truth with which each of the weak classifiers and the final strong classifier is compared against to calculate the accuracy.

C. Experiments and Results

We design two experiments to evaluate our approach: intra-subject and inter-subject experiments. In the intra-subject experiment, we train and test on the same subject. The intra-subject strong classifiers are built for each subject and interaction. 80% of a subject's data is used to train a strong classifier and 20% is used for testing. The data window size that gave the best accuracy for each interaction was determined empirically. A smoothing operation using a moving average filter was applied to the results of each strong classifier to smooth out irregularities in the results. Table 1 shows the average accuracy of the intra-subject strong classifiers and the average accuracy of the best weak classifiers for all interactions. The averaging of the accuracies was performed over all five subjects. These accuracies are termed window-by-window accuracies since a decision is taken for each data window within a data collection. Apart from this we also report the instance detection accuracy. One interaction instance is the whole duration of an interaction. For example, if a person performs *Picking Up Cup* for 3 seconds, there are several data windows within the interaction. However this is counted as only one instance of the interaction. Instance detection accuracy shows the number of times each strong classifier was able to correctly identify the instances of the interaction compared to the number of times the interaction actually occurred in the data collection.

TABLE 1. INTRA-SUBJECT CLASSIFIER ACCURACIES

Interaction	Instance Detection Accuracy	Accuracy Avg.	Best Weak Classifier Avg.
Picking Up Cup	100%	86.1%	78.1%
Carrying Backpack	80%	86.2%	78.6%
Pushing-Pulling a Chair	85%	70.7%	67.8%
Writing with a pen	100%	86.9%	75.9%
Shaking Hands	100%	88.4%	82.5%
Walking Together	100%	81.9%	76.3%

The amount of improvement afforded by the AdaBoost algorithm varies from interaction to interaction. From Table 1 it can be seen that, in half the interactions, the AdaBoost classifier increases upon the accuracy of the best weak classifiers by around 10%. The accuracy gain for *shaking hands*, *pushing-pulling a chair, and walking together* interactions are not as high (approximately 5%). However, the AdaBoost algorithm creates a classifier that is stronger than the individual weak classifiers and provides more confidence in detection than a weak classifier. We can also see that all of the classifiers detect interaction instances very well. The window-by-window accuracy is lower than the instance detection accuracy because it accounts for false positives and false negatives.

In the inter-subject experiment, leave one subject out technique is applied for every interaction. For data collection involving five subjects, a strong classifier is trained on data from four subjects and tested on the fifth subject's data. This training and testing is repeated five times to generate five strong classifiers. Each time we select a different subject's data for testing the classifier and the remaining four for training the classifier. Here again the data window size for each interaction was chosen empirically and the classifier results were filtered using a smoothing operation. The accuracy average for each interaction is obtained by averaging the accuracies over all inter-subject classifiers for that interaction and is shown in Table 2. This accuracy represents the window-by-window accuracy. Table 2 also shows the instance detection accuracy which is calculated for all the strong classifiers built for inter-subject verification for an interaction.

TABLE 2. INTER-SUBJECT CLASSIFIER ACCURACIES

Interaction	Instance Detection Accuracy	Accuracy Avg.
Picking Up Cup	98%	76.8%
Carrying Backpack	84%	84.2%
Pushing-Pulling a Chair	58%	69.6%
Writing with a pen	90%	75.6%
Shaking Hands	94%	76.8%
Walking Together	96%	79.7%

From Table 2 it can be seen that the inter-subject classifier generated by AdaBoost can detect most interactions with reasonably good accuracy. All interactions except *pushing-pulling a chair* have an accuracy above 75%. For the *pushing-pulling a chair* even the intra-subject classifier had a lower accuracy than the other intra-subject classifiers. Perhaps the accuracy is lower than expected because the operators that we used to detect similar movements are ill-equipped to detect the "sliding" motion that the sensor on the chair experiences and map it to the push/pull of the hand, when the chair is pushed or pulled. This is reflected in the instance detection accuracy for *pushing-pulling a chair* as well. It can also be seen that the window-by-window accuracies are lower than the interaction detection accuracies because they also account for false positives and false negatives.

VII. CONCLUSION AND FUTURE WORK

In this paper, we explored how IMUs can be utilized to detect human-human/object interactions using several common time-domain signal processing techniques. The results show that all of the interactions are reasonably detectable by a combination of the operators and the boosting algorithm. Looking at the intra-subject results, the important point is that the average accuracy of the AdaBoost classifier for each of the interactions is better than the average accuracy of the best weak classifier for that interaction. Thus, the fusion of the weak classifiers leads to a classifier that is better than the sum of its parts. In the future, we will investigate more interactions that have two or more actors. We will also investigate other learning algorithms that are better adapted to recognize the features best representing an interaction. Because interactions vary so much and are numerous, it is infeasible to find the best operators for each interaction. Therefore, it will also be constructive to investigate methods that systematically choose the best operators for interactions automatically.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation, under grants CNS-1150079 and CNS-1012975, and the TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, pp. 304–307, Springer, 1999.

[2] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85–90, IEEE, 1994.

[3] H. M. Hondori, M. Khademi, and C. V. Lopes, "Monitoring intake gestures using sensor fusion (microsoft kinect and inertial sensors) for smart home tele-rehab setting," in *2012 1st Annual IEEE Healthcare Innovation Conference*, 2012.

[4] O. Yürüten, J. Zhang, and P. H. Pu, "Predictors of life satisfaction based on daily activities from mobile sensor data," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 497–500, ACM, 2014.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.

[6] J.-y. Hong, E.-h. Suh, and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

[7] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 1, pp. 1–12, 2010.

[8] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, *et al.*, "The mobile sensing platform: An embedded activity recognition system," *Pervasive Computing, IEEE*, vol. 7, no. 2, pp. 32–41, 2008.

[9] R. E. Schapire, "Explaining adaboost," in *Empirical inference*, pp. 37–52, Springer, 2013.

[10] M. J. Marn-Jiménez, R. Muñoz-Salinas, E. Yeguas-Bolivar, and N. P. de la Blanca, "Human interaction categorization by using audio-visual cues," *Machine vision and applications*, vol. 25, no. 1, pp. 71–84, 2014.

[11] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from rgbd images," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 842–849, IEEE

[12] E. M. Tapia, S. S. Intille, and K. Larson, *Activity recognition in the home using simple and ubiquitous sensors*. Springer, 2004.

[13] C. Qu, D. Zhang, Y. Lin, and S. Wang, "Distant human interaction recognition with kinect," *Journal of Computers*, vol. 9, no. 9, pp. 2091–2099, 2014.

[14] B. Ur, E. McManus, M. Pak Yong Ho, and M. L. Littman, "Practical trigger-action programming in the smart home," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 803–812, ACM, 2014.

[15] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Ambient assisted living and home care*, pp. 216–223, Springer, 2012.

[16] A. Z. Zivotofsky and J. M. Hausdorff, "Journal of neuroengineering and rehabilitation," *Journal of Neuroengineering and Rehabilitation*, vol. 4, p. 28, 2007.

[17] R. Lotfian and R. Jafari, "An ultra-low power hardware accelerator architecture for wearable computers using dynamic time warping," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 913–916, EDA Consortium, 2013.

[18] H. Junker, P. Lukowicz, and G. Troster, "Sampling frequency, signal resolution and the accuracy of wearable context recognition systems," in *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, vol. 1, pp. 176–177, IEEE, 2004.

[19] T. R. Bennett, C. Savaglio, D. Lu, H. Massey, X. Wang, J. Wu, and R. Jafari, "Motionsynthesis toolset (most): a toolset for human motion data synthesis and validation," in *Proceedings of the 4th ACM MobiHoc workshop on Pervasive wireless healthcare*, pp. 25–30, ACM, 2014.