Gesture recognition using Symbolic Aggregate approximation and Dynamic Time Warping on Motion Data

Antigoni Mezari
University of Piraeus
Dept. of Digital Systems
Piraeus, Greece
antigoni.mezari@gmail.com

Ilias Maglogiannis
University of Piraeus
Dept. of Digital Systems
Piraeus, Greece
imaglo@unipi.gr

ABSTRACT
In the area of advanced human-computer interaction, automatic gesture recognition is an important field. Motion data produced by the accelerometer of a smart watch can be utilized in hand gesture recognition. In this work we examine the use of a commodity smart watch and a smartphone as the capture and the processing units respectively, for recognizing gestures. We claim that if the proper gesture recognition algorithms are applied, the recognition of natural gestures i.e. 3-D gestures easily performed by an individual can be accurate enough to be useful in everyday life activities. Symbolic Aggregate Approximation (SAX) and Dynamic Time Warping (DTW) methodologies are utilized in this context and evaluated using a set of six 3-D natural gestures.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation): User Interfaces - Input devices and strategies; I.5.2 Pattern Recognition: Design Methodology - Classifier design and evaluation.

Author Keywords
Gesture recognition; accelerometry motion sensor; SAX; DTW; smart watch; Android; Pebble.

INTRODUCTION
Gesture recognition is considered extremely important for smart and efficient human computer interaction. It is a key component in various application domains, including assisted living [11, 21], activity recognition [3, 14, 15, 17], smart training and coaching [16], and others. Gesture recognition can be realized in different ways, using devices such as cameras or other sensors to track motion. Currently, recognition using the visual signal is the most widespread general approach to recognizing gestures [12, 10]. In such approaches, the cameras are usually installed at fixed locations, and gesture recognition is possible only in a confined, pre-specified space.

PervasiveHealth '17, May 23–26, 2017, Barcelona, Spain

ACM ISBN 978-1-4503-6363-1/17/05…$15.00
https://doi.org/10.1145/3154862.3154927

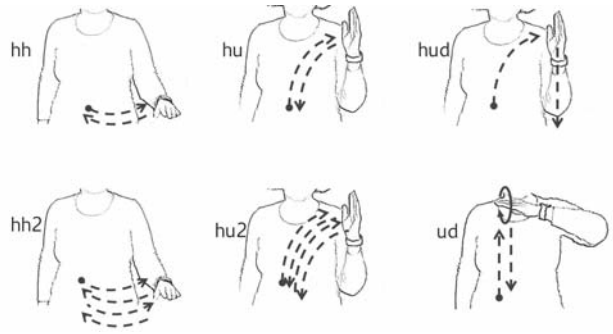

Figure 1. The selected gestures, a dot depicts a gesture start.

An alternative to vision-based approaches is to use wearable motion sensors, like a smart watch. A smart watch equipped with accelerometer can provide information about the movement of the hand that may be used for recognizing gestures. The main advantage of using a smart watch is that it doesn't impose restrictions to the user. For example, it could be used for controlling a home appliance (e.g. sound system, air-condition), or lighting, from everywhere in the house, without requiring camera-based monitoring of all rooms. Furthermore, it could be used for navigating within virtual environments (e.g. a VR game or educational application). In all cases, the use of a smart watch eliminates the need for special-purpose sensors or privacy-compromising devices (such as cameras), which is particularly advantageous when involving seniors or people with disabilities.

The aim of this work is to examine whether simple and natural gestures, such as those depicted in Figure 1, can be reliably recognized using commodity devices like a smart watch and a smartphone, and to propose specific methodologies to improve performance and accuracy.

Related work and background information
Some of the techniques proposed for gesture recognition are designed especially for rapid prototyping of gesture-based user interfaces. The $1 recognizer [19], a 2-D unistroke recognizer, is one of them. Its preprocessing algorithm consists of 4 steps; namely resampling, rotation, scaling, moving. Initially the gesture is resampled into a fixed number of points, evenly spaced along the path. Then the path is rotated so that the line from the centroid of the path to the first point of the

path is parallel to the x-axis. Non-uniform scaling is used to fit the path in a reference square. Finally the path is moved so that its centroid matches (0,0). To compare the gesture with the templates, the mean value of the Euclidean distance between the corresponding points is computed. In addition to the above mentioned preprocessing steps the authors propose the use of an iterative method to fine-tune the rotation angle. The gesture is classified to the template it has the minimum distance from. Another recognizer quite similar to $1 is Protractor [5]. It uses cosine distance to calculate the distance between two gestures, and calculate the rotation angle so that the cosine distance between the two gestures is minimized. The gesture is not scaled. The $N recognizer [1] is a modification of $1 intended for the recognition of multistroke 2-D gestures. $N-protractor [2] is a variation of $N embedding the technique of Protractor. The $P recognizer [18] represents gestures as unordered point-clouds. The $3 recognizer [4] is a variation of the $1 technique intended to manipulate 3-D data recorded using the Wii Remote device. Another 3-D gesture recognizer is uWave [8]. This method uses the data of a three-axis accelerometer. During preprocessing the data are compressed by an averaging window and subsequently the new values are non-linearly quantized. Dynamic Time Warping (DTW) is used to match two time series and the Euclidean distance is used for distance calculation. The evaluation in [8] was again performed on data captured with the Wii Remote device. Xie et al [20] examine gesture recognition using a smartphone as the sensor. They use dynamic-threshold truncation to remove data recorded before the gesture actually starts and after the gesture ends. They apply a low-pass filter to smooth the data. To this time series, they further append the amplitudes of its Fast Fourier Transform.

In contrast to previous works, gesture recognition using a commodity smart watch as the motion sensor is examined here. The Pebble smart watch features a 3-axis accelerometer. It is calibrated to measure a maximum acceleration of $\pm$4G (which was found during our experimental evaluation to be sufficient for capturing correctly all the performed gestures) and produces integer data measured in milli-Gs. It communicates with an Android or IOS device using the Bluetooth 4.0 (Bluetooth Low Energy) protocol. An abrupt movement of the Pebble produces a tap event. The user can perform such an abrupt movement of their wrist to mark the start and the end of a gesture.

GESTURE RECOGNITION METHODOLOGY

Selected gestures, measurements and recognition methods

The gestures used for the evaluation of the proposed methodology were selected according to the following criteria: they are characterized by the wrist movement, they are simple and natural gestures and they are easily repeated, they are different from each other in accordance to the combination of the orientation of the watch and the direction of its movement, the gravity does not complicate their recognition and if possible contributes to their differentiation, and they can be related to commands for the manipulation of devices. The selected set is shown in Figure 1. Nevertheless, the proposed system is flexible and it can be trained to any kind of gesture dataset.

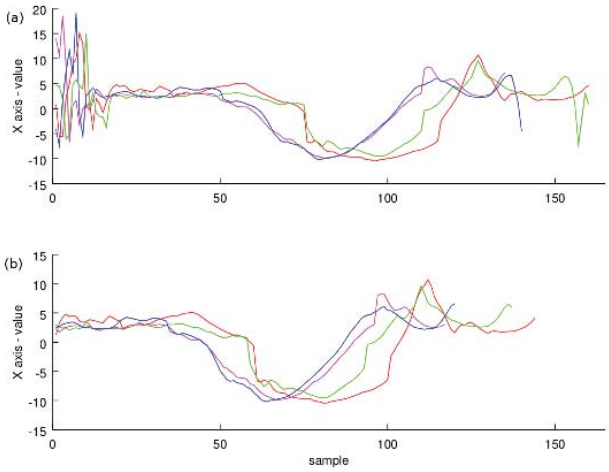

Figure 2. One gesture performed four times by the same user. (a) Raw data. (b) Data after removing the noise of the tap event.

The three axes along which acceleration is measured are bound to the watch. The accelerometer data received during a gesture include gravity and are also affected by the change of the orientation of the smart watch during the movement. Data also contain noise. The use of tap event at the beginning and the end of the gesture affects the accelerometer data measurements at those points. Since the duration of a gesture varies and the accelerometer collects data at a constant rate, the number of samples for each gesture differs. A heuristic algorithm to eliminate the effect of the tap event was used. Starting from the thirtieth measurement from the start and till the thirtieth measurement to the end, the maximum difference between two successive measurements was determined, in order to be used as a threshold. Then, starting from the thirtieth measurement from the start and moving towards the first measurement, the difference between two successive measurements is calculated; if it exceeds the threshold value, the part of the gesture before that point is excluded. A similar procedure is applied to exclude a part of the time series near the end of it. The resulting time series that represent a single gesture are then used as input to any of the three techniques for gesture recognition, which are discussed in the following subsections. Figure 2 illustrates the measurements received along the X-axis when a user performed the same gesture four times. The effect of tap event is obvious in (a) and it is removed in (b).

Figure 3 illustrates the workflow pipeline of the proposed methods; the SAX, the DTW and the SAX-DTW method, presented in the next subsections. In all methods the last step involves the classification of an incoming, unknown gesture against a “training set” of known gestures. A simple classifier such as the k-nearest neighbor algorithm with k=1, along with the distance function defined by each of the tree evaluated methods, is used for this purpose.

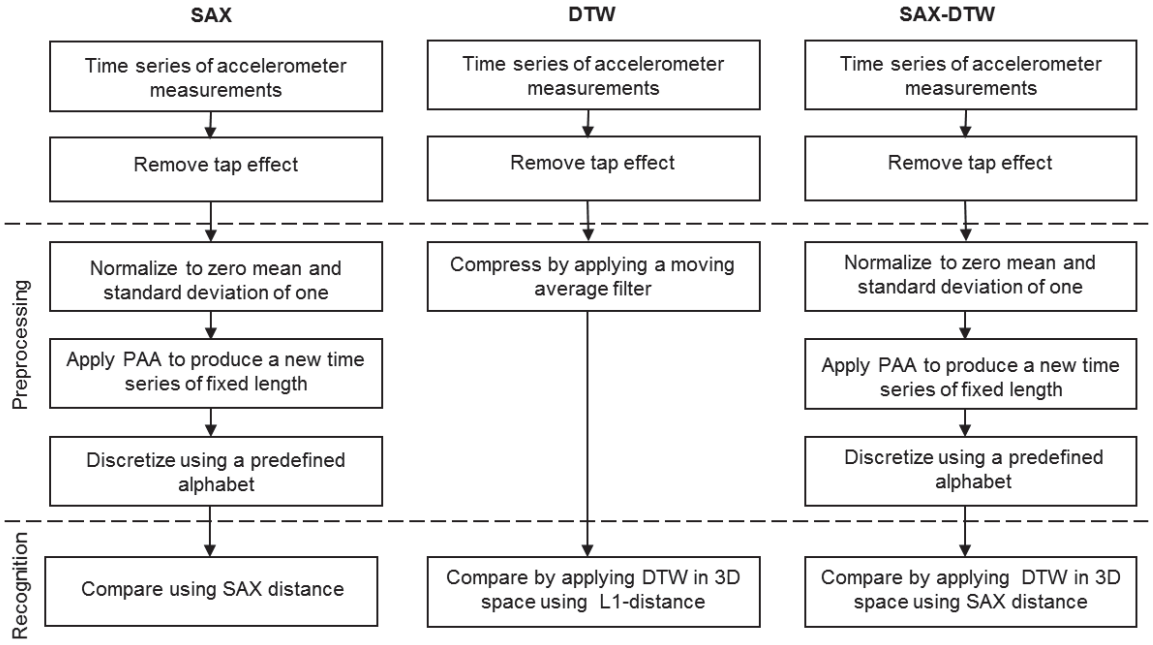

Figure 3. The main steps of the SAX, DTW and SAX-DTW methods.

The SAX method

SAX (Symbolic Aggregate approXimation) [6, 7, 9] transforms a time series of arbitrary length n to a string of length w using an alphabet $A = \{a_1, a_2, ...a_I\}$. The time series is initially normalized to have a mean of zero and a standard deviation of one. Then, Piecewise Aggregate Approximation (PAA) is used to reduce the dimensionality from n to w. Subsequently, the "breakpoints" b_i, $i = 1, ...I-1$, to be used for the discretization of the transformed time series, are selected according to [6]. The time series is discretized to the selected alphabet by replacing each value of it that lies between breakpoints b_i and b_{i+1} (for $i = 1, ...I-2$) with symbol a_{i+1} (reserving symbols a_1 and a_I for values lower than b_1 and higher than b_{I-1}, respectively). The distance between two symbols is defined to be zero if their index differs at most by one (i.e. is zero between symbols a_i and a_{i+1}). In any other case, the distance between symbols a_i and a_k, where $k > i$, is defined as equal to $b_{k-1} - b_i$. The distance between two strings (each corresponding to a different time series and comprising w symbols) is computed as the mean of their pair-wise symbol distances (i.e. the mean of the distance between the first symbol of each of the two strings, the distance between the second symbol of each of the two strings, and so forth).

To apply the SAX method, we initially combined the data of the three axes in one time series, first all the x-axis data followed by all the y-axis data and the z-axis data. We then normalized each time series to have a mean of zero and a standard deviation of one, thus jointly normalizing the x, y and z axis data. The parameters of the SAX method we used were $w = 96$ and $I = 7$, which means we selected to represent a gesture with a string of 96 symbols (32 for each axis) using an alphabet of 7 symbols. These parameter values were chosen based on preliminary experiments. Figure 4 illustrates the final time series produced by the SAX method to represent the same gesture performed by two different users. The corresponding symbols are connected.

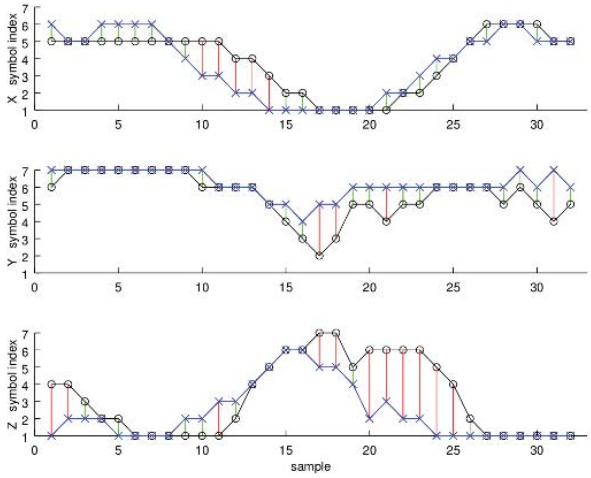

Figure 4. Two strings produced by the SAX method representing the same gesture performed by two different users. Only the pairs for which the indexes of their symbols differ by more than one contribute to the distance between the two strings.

The complexity of the preprocessing algorithm is $O(n)$, where n is the length of the time series of the gesture measurements. The complexity of the recognition algorithm is $O(n)$, where n is the parameter w of the SAX method.

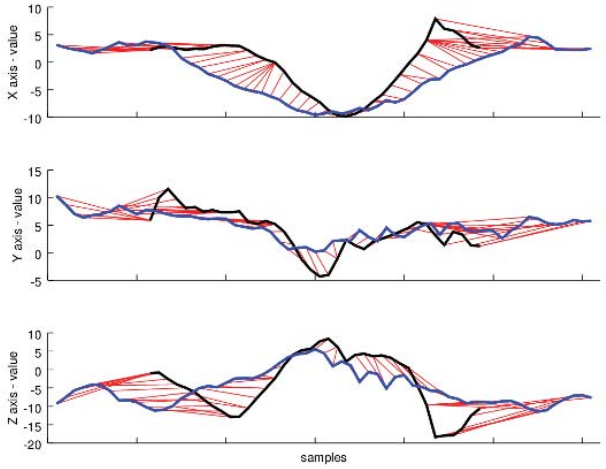

Figure 5. Two time series that the DTW method produced after compression for the same gesture performed by two users. The corresponding points are connected.

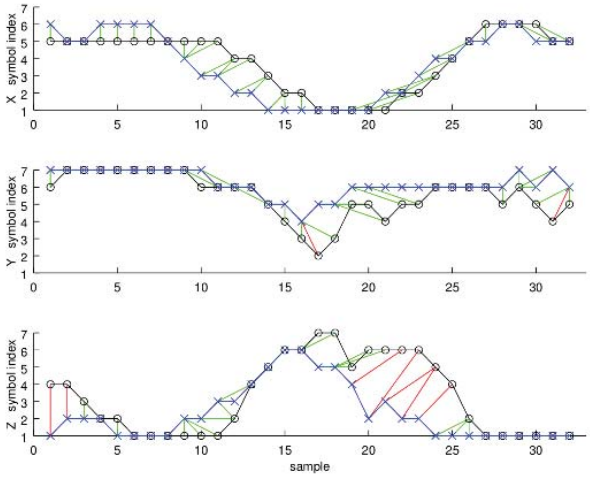

Figure 6. According to the SAX-DTW method, only the symbol pairs that their symbol indexes differ more than one contribute to the distance between the two strings.

The DTW method

Dynamic time warping (DTW) [13] is an algorithm for assessing the similarity of two signals (time series), which is particularly useful when signals of similar form can differ in the speed of their evolution in time (e.g., time series corresponding to the same gesture, where a part of that gesture is executed at a different speed each time). To apply DTW, we first compress the original time series by applying to it a moving average filter of width equal to 4 samples, with a sliding step equal to 3 samples (thus limiting the length of the compressed time series to one third of the original one). In applying DTW to a pair of time series, we also adopted the Sakoe-Chiba constraint [13], which restricts the potentially matching samples of the two time series within an "adjustment window". The width of this adjustment window was set to 7 in our experiments. The distance between two points of the pair of time series is calculated in the 3D space, i.e. jointly considering the measurements in all three axes of the accelerometer, using the L1 distance. Then, the distance between the two time series is calculated as the sum of pairwise absolute differences for all the sample pairs dictated by the DTW result, divided by the length of the time series that we want to classify. Figure 5 illustrates the final time series produced by the DTW method to represent the same gesture performed by two different users and the corresponding points of them.

The complexity of the recognition algorithm is $O(T * max(n,m))$ where T is the width of the adjustment window and n, m are the lengths of the two compared time series.

The SAX-DTW method

The coexistence of SAX and DTW proposed in this work combines the best characteristics of both methods, i.e. the effectiveness and low complexity of SAX and the insensitivity of DTW to speed fluctuations during the execution of a gesture. For this reason, the proposed combination adopts the alphabet-based time series representation of SAX, and the definition of the distance between two symbols of the same method. But, instead of using the default pairwise comparisons of SAX, i.e. comparing the 1st, 2nd etc. symbols of the two strings, it treats the two strings as input to the DTW method (as described above, with the Sakoe-Chiba constraint), thus allowing DTW to find the optimal "warping" of the second string to the first one. Then, the distance between the two strings is calculated as the sum of pairwise symbol similarities for all the symbol pairs dictated by the DTW result, divided by the length of the string. Figure 6 illustrates the use of the SAX-DTW method.

The complexity of the recognition algorithm is $O(T * n)$ where T is the width of the adjustment window and n is the parameter w of the SAX method.

SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

Gesture recognition system implementation

The implemented system consists of two companion applications. The first application runs on the smart watch (Pebble) and is responsible for capturing the accelerometer measurements and sending them to the Android application. The second application runs on the companion Android device. The Android application provides the interface to the user, receives the motion data from the smart watch, updates the database (template library) with the training gestures and runs the recognition methods. Pebble and the Android device communicate with each other using Bluetooth. The combination of the two developed applications allows the person wearing the smart watch to perform a gesture and, immediately after the second tap event (which marks the end of a gesture), to have the gesture recognized.

Method	Correct classification (%) Average (all sets)	Worst (one set)
SAX	99.01	94.44
DTW	97.04	83.33
SAX-DTW	**99.67**	**98.00**

Table 1. Results of personalized gesture recognition.

Recognized as	Performed gesture hh	hu	hud	ud	hh2	hu2
hh	48					
hu		45				
hud			53			
ud				54		
hh2	1				52	
hu2						51

Table 2. Confusion matrix of personalized gesture recognition using SAX-DTW.

Method	Correct classification (%) Average (all sets)	Worst (one set)
SAX	96.44	86.11
DTW	93.15	72.22
SAX-DTW	**99.21**	**97.14**

Table 3. Results of user-independent gesture recognition.

Recognized as	Performed gesture hh	hu	hud	ud	hh2	hu2
hh	118				1	
hu		112				
hud		1	131			
ud				134		
hh2	4				129	
hu2						129

Table 4. Confusion matrix of user-independent gesture recognition using SAX-DTW.

Method evaluation

In order to evaluate the application and the utilized methods we asked four persons to perform the gestures illustrated in Figure 1. Each set included at least 5 repetitions of the same gesture. A few days later the same persons performed additional gesture sets. Altogether, we collected 8 sets with a total of 304 gestures. Two types of evaluation were performed in order to assess the robustness of the proposed methods: one involving training by the same person and a second utilizing training by different users.

User-dependent recognition (personalized gesture recognition)

In this case the first gesture set of user A was used to train the system and the second set of the same user for testing. This process was repeated using the second set to train the system and the first for testing. The above mentioned process was repeated for all users and the average values are reported in Table 1 for the three recognition methods. The last column of this table reports the results for the gesture set where the corresponding method performed the worst; this is an indication of how much the performance can be expected to degrade for a single user. The SAX-DTW method outperforms all the other methods, achieving correct classification rate that is on average equal to 99.67% of the input gestures, and ranges for the individual sets from 98.00% to 100%. Table 2 shows the confusion matrix for the SAX-DTW method.

User-independent recognition

In order to evaluate the system in the case of the user-independent recognition, the following process was applied. The first set of user A was used to train the system. All the gestures of the other users (B, C and D) were used as the gestures to be recognized. The same process was repeated three times using the first set for the other users (B, C and D) as the training set. The results are summarized in Table 3. The SAX method produced the correct response in 96.44% of the occasions on average, with the lowest rate being 86.11%. The DTW method produced correct results in 93.15% of the occasions on average, with the lowest correct classification rate being 72.22%. The SAX-DTW method outperforms all the other methods, achieving correct classification rate that is on average equal to 99.21% of the input gestures, and ranges for the individual sets from 97.14% to 100%. Table 4 shows the confusion matrix for the SAX-DTW method.

CONCLUSIONS

In this work we examined the feasibility of a gesture recognition system based on a smart watch and a connected smart phone. The motion measurements obtained by the accelerometer of the smart watch during a gesture were captured and relayed to the smart phone. Three alternative recognition methods were implemented. The proposed SAX-DTW method was shown to outperform the other two examined methods and to produce highly accurate results, even in the case of user-independent training.

Future work will include testing the proposed gesture recognition technique in specific applications in the domain of health and wellbeing, such as human-computer interaction for people with disabilities. A field of interest to apply the SAX-DTW method is automatic sign language recognition.

ACKNOWLEDGMENT

Ilias Maglogiannis wish to acknowledge the support received from University of Aegean Research Committee for conducting this research.

REFERENCES

1. Lisa Anthony and Jacob O Wobbrock. 2010. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*. Canadian Information Processing Society, 245–252.

2. Lisa Anthony and Jacob O Wobbrock. 2012. $N-Protractor: A fast and accurate multistroke recognizer. In *Proceedings of Graphics Interface 2012*. Canadian Information Processing Society, 117–120.

3. Jonathan Knighten, Stephen McMillan, Tori Chambers, and Jamie Payton. 2015. Recognizing social gestures with a wrist-worn smartband. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 544–549.
4. Sven Kratz and Michael Rohs. 2010. A $3 gesture recognizer: simple gesture recognition for devices equipped with 3D acceleration sensors. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 341–344.
5. Yang Li. 2010. Protractor: a fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2169–2172.
6. Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2–11.
7. Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15, 2 (2007), 107–144.
8. Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675.
9. Ayaka Onishi and Chiemi Watanabe. 2011. Event Detection using Archived Smart House Sensor Data obtained using Symbolic Aggregate Approximation. PDPTA.
10. Pramod Kumar Pisharady and Martin Saerbeck. 2015. Recent methods and databases in vision-based hand gesture recognition: A review. *Computer Vision and Image Understanding* 141 (2015), 152–165.
11. Lorenzo Porzi, Stefano Messelodi, Carla Mara Modena, and Elisa Ricci. 2013. A smart watch-based gesture recognition system for assisting people with visual impairments. In *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices*. ACM, 19–24.
12. Siddharth S Rautaray and Anupam Agrawal. 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review* 43, 1 (2015), 1–54.
13. Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.
14. Sougata Sen, Kiran K Rachuri, Abhishek Mukherji, and Archan Misra. 2016. Did you take a break today? Detecting playing foosball using your smartwatch. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*. IEEE, 1–6.
15. Sougata Sen, Vigneshwaran Subbaraju, Archan Misra, Rajesh Krishna Balan, and Youngki Lee. 2015. The case for smartwatch-based diet monitoring. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 585–590.
16. Kosuke Takano, Kin Fun Li, and Mark Graham Johnson. 2011. The design of a web-based multimedia sport instructional system. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*. IEEE, 650–657.
17. Edison Thomaz, Irfan Essa, and Gregory D Abowd. 2015. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1029–1040.
18. Radu-Daniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. 2012. Gestures as point clouds: a $P recognizer for user interface prototypes. In *Proceedings of the 14th ACM international conference on Multimodal interaction*. ACM, 273–280.
19. Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 159–168.
20. Michael Xie and David Pan. 2014. Accelerometer Gesture Recognition. (2014).
21. Chun Zhu and Weihua Sheng. 2011. Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41, 3 (2011), 569–573.