Continuous Gesture Recognition for Resource Constrained Smart Objects

Bojan Milosevic, Elisabetta Farella, Luca Benini
DEIS - Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Bologna, Italy
{bojan.milosevic, elisabetta.farella, luca.benini}@unibo.it

Abstract — **Tangible User Interfaces (TUIs) feature physical objects that people can manipulate to interact with smart spaces. Smart objects used as TUIs can further improve user experience by recognizing and coupling natural gestures to commands issued to the computing system. Hidden Markov Models (HMM) are a typical approach to recognize gestures sampled from inertial sensors. In this paper we implement a HMM-based continuous gesture recognition algorithm, optimized for low-power, low-cost microcontrollers without floating point unit. The proposed solution is validated on a set of gestures performed with the Smart Micrel Cube (SMCube), which embeds a 3-axis accelerometer and an 8-bit microcontroller. Through the paper we evaluate the implementation issues and describe the solutions adopted for gesture segmentation and for the fixed point HMM forward algorithm. Furthermore, we explore a multiuser scenario where up to 4 people share the same device. Results show that the proposed solution performs comparably to the standard forward algorithm and can be efficiently used for low cost smart objects.**

Keywords — Hidden Markov Models; Tangible Interfaces; Smart Objects; Gesture Recognition; Fixed Point.

I. INTRODUCTION

Tangible User Interfaces (TUIs) introduce physical, tangible objects that augment the real physical world by coupling digital information to everyday objects. The system interprets these devices as part of the interaction language. TUIs become the representatives of the user navigating in the environment and enable the exploitation of digital information directly with his/her hands. People, manipulating those devices, inspired by their physical affordance, can have a more direct access to functions mapped to different objects.

The effectiveness of a TUI can be enhanced if we use sensor augmented devices, which can provide a bridge between the physical and the digital world. Such *smart objects* may be able to recognize user gestures and improve human experience within interactive spaces. Furthermore, the opportunity to execute on-board a gesture recognition algorithm, without the need to end data streams from the local sensor to a central base station, results in *extended battery life*, improved system *scalability* and easier handling of *mobile TUIs*.

In this work we present an algorithm for segmentation and gesture recognition implemented on-board of a smart object, the *Smart Micrel Cube* [4], which embeds an 8 bit microcontroller, a digital accelerometer and a Bluetooth transceiver. This device can be used as the tangible interface of an interactive tabletop setup (like in the TANGerINE Project [4]) or as a mobile context-aware interface towards a smart, environment [6]. The algorithm detects the beginning and the end of motion segments and uses Hidden Markov Models to recognize the executed gesture. Unlike our implementation, gesture segmentation from a continuous stream of inertial data often relies on user collaboration (e.g., pushing a button wile executing a gesture [11]) or integrates information from various types of sensors (e.g., ultrasonic [19], microphones [21]). HMMs have been broadly applied to gesture recognition [11], [14], [15] but implementation on low performance devices are limited to high resource mobile-devices and 32 bit microcontrollers [2]. We focused on a resource constrained platform and addressed implementation issues for a 8 bit fixed point microcontroller.

The rest of the paper is organized as follow: Section II reports on related works and the sub-sequent Section III describes the system and the recognition procedure. Following, we characterize our implementation in Section IV; discuss experimental analysis and results in Section V and we conclude our paper in Section VI.

II. RELATED WORK

The use of TUIs has been proposed in many scenarios where users manipulate digital elements. This have been proved to be useful especially in applications for entertainment and education [16], exploration of virtual environments [9], media content creation and manipulation [10], [18]. The entertainment market is rapidly embracing tangible and gestural interfaces in several new scenarios, as for game-console controllers, such as the Wii, or for mobile devices and smart phones.

Smart objects with gesture recognition capabilities can enhance the expressiveness of TUIs. The MusicCube, for example, is a tangible interface used to play digital music like an MP3 player [5]. The cube is able to understand the face pointing upward and a set of simple gestures. This ability, together with a set of controls and buttons, is used to choose the desired playlist and to control music volume.

Gestures executed with natural hand and arm movements are variable in their spatial and temporal execution, requiring classifiers suited for temporal pattern recognition. Typical approaches include Dynamic Time Warping (DTW) [13],

 ISBN: 978-1-61208-100-7

Neural Networks [3], and Hidden Markov Models (HMMs). HMMs are often used in activity recognition since they tend to perform well with a wide range of sensor modalities and with temporal variations in gesture duration. They are also used successfully in other problem domains, such as speech recognition, for which they were initially developed [17]. Several variants of HMMs have been proposed to recognize inertial gestures: in [11] 5-state ergodic discrete HMMs are evaluated with the Viterbi algorithm to classify gestures performed with a handheld sensor device in several tasks (interaction with a TV, a presentation or a CAD environment). The work of Mantyla et al. [15] uses 7-states Left-to-Right models and the forward algorithm to classify actions performed with a mobile phone equipped with an accelerometer. Both implementations have similar performance and rely on a PC to execute all computations. In our work we are using low-power hardware without a floating point unit, so we implemented a fixed-point variant of the forward algorithm, presented in a previous work [22].

Using HMMs to classify gestures from a continuous stream of data brings another issue to solve: the recognition procedure needs to discriminate actually executed gestures from all the other arbitrary movements. Hoffman et al. [8] use a sensorized glove to recognize hand gestures: to segment the data stream they compute the velocity profile of the sampled accelerations and apply a threshold to identify the motion segments. In [7] a Gaussian model of the stationary state is used with a sliding window approach to find pauses in movements, which identify the beginning and the end of a gesture. Amft et al. [1] presented an algorithm to recognize arm activity during meal intake, with accelerometers placed on the arm and the wrist of the user. To segment gestures they use the Sliding Window and Bottom-up (SWAB) algorithm [12] and the angle of the lower arm as the segmentation feature. While those works have focused to develop recognition solutions, none of them deals with computation or memory limited devices. We found a similar solution implemented on a wristwatch device, using a 32 bit ARM microcontroller [2], but there are no works targeting low-cost, low-power 8 bit microcontrollers, such is the Atmel ATmega168 used in this work.

III. SYSTEM OVERVIEW

The smart object used in this work is a cube shaped artifact, the Smart Micrel Cube (SMCube) illustrated in Fig. 1. It embeds a low-cost, low-power 8-bit microcontroller (Atmel ATmega168), a Bluegiga WT12 Bluetooth transceiver, which supports Serial Port Profile (SPP) and a MEMS tri-axial accelerometer (STM LIS3LV02DQ) with a programmable full scale of 2g or 6g and digital output. The cube is powered through a 1000 mA/h, 4.2 V Li-ion battery. With this battery the cube reaches up to 10 hours of autonomy during normal operation.

The processing flow is illustrated in Fig. 2. Accelerations on the three axes are sampled at a rate of 31.75 Hz within the range of ±2g. The accelerometer represents the sampled data with a 16 bit integer value, and reaches a resolution of 1 mg.

In the pre-processing stage, sampled data are filtered with an averaging filter to eliminate high frequency noise. This filter computes the average value of the last 4 samples: this window

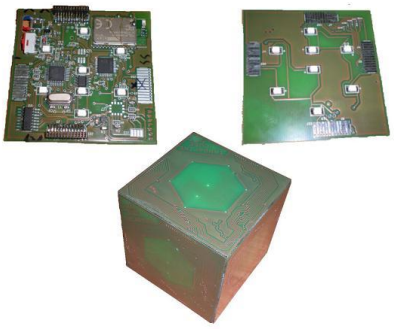

Figure 1. Smart Micrel Cube: on the top left the inner surface of the master face, with all the main components and on the top right the inner surface of the other faces

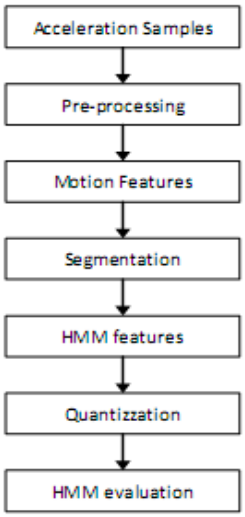

Figure 2. System processing blocks

length was chosen to use simple shift operations instead of divisions. An offset filter removes the stationary gravity acceleration, measured during a calibration phase which consist in sampling data when the device is placed still on a table.

The next two stages implement a motion detection algorithm, used for segmentation. For this purpose two features are used: delta, which consists in the difference between the actual sample and the mean of the last 4 samples, and the variance. A Finite State Machine (FSM) uses these features to find out if the device is in one of the 4 possible states: still on a table, still in user's hand, in movement, shaken. We asked users to hold still in a hand the device right before and after executing a gesture. In this way we segment as gestures only motion segments which start and end with this particular condition and have a limited duration.

Each acceleration vector of a gesture, $\{a_x, a_y, a_z\}$ is converted to equivalent spherical coordinates $\{\varphi, \theta, r\}$, as represented in Fig. 3. From those vectors, magnitude information r is discarded and the angles φ and θ are used to identify the direction of the movement performed, represented

 ISBN: 978-1-61208-100-7

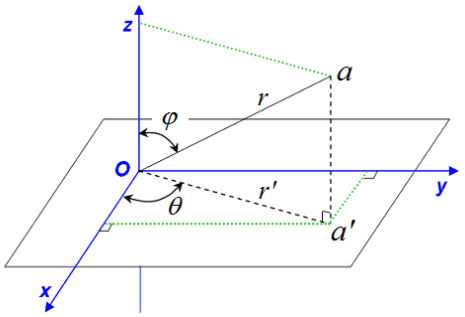

Figure 3. Spherical coordinates

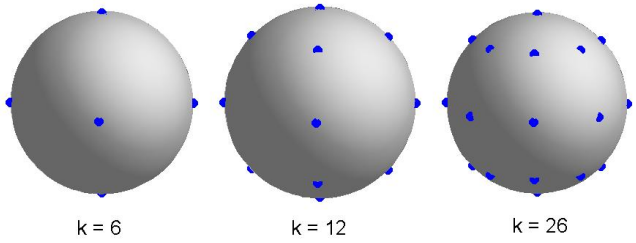

Figure 4. Codebook vectors for direction quatization.

as a point on a unitary sphere. In order to cluster this data for the discrete HMMs, a quantization algorithm is applied, with *k* vectors or codes in the codebook. In this case, codebook vectors are uniformly distributed points on the unitary sphere, as illustrated in Fig. 4. Since *k* must be determined empirically we decided to conduct tests to find a codebook size delivering a satisfying trade-off between results and algorithm complexity.

The last block of the processing chain is the HMM classifier. During the training phase we collected several gestures to build and validate models for each gesture. The training was implemented on a PC with Matlab and the HMMtoolbox, which uses the well-known Baum-Welch algorithm. We chose to use discrete HMMs, with 7-state Left-to-Right models for all gestures, according to the results of a preliminary exploration. For the on-line recognition a modified fixed-point version of the forward algorithm was implemented in both Matlab and C on the AVR platform.

IV. IMPLEMENTATION

The goal of this work is to implement the whole gesture recognition algorithm on board of the SMCube. The main tasks performed are the segmentation of gestures from a continuous data stream and the HMM-based gesture recognition.

A. Segmentation

The segmentation algorithm was implemented ad-hoc, and even if it was developed and optimized in this particular setup, it can be used in similar scenarios.

It is not possible to recognize gestures with only one accelerometer, if they are part of a larger continuous movement. To overcome this problem we added a limitation in gesture execution: users must hold still the device for few instants before and after a gesture. In this way, it is fundamental for the algorithm to identify when the device is still in user's hand and when a movement starts and ends. To evaluate the state of the device we compute the variance of the filtered signal and use a FSM. The variance uses sliding windows of 4 samples; it is calculated for each axis and then summed to have a total information of the intensity of the movement.

When the cube is placed still on a surface, the variance values observed are near zero. If a user holds the device in a hand, we measure a low and uniform variance, always within a limited interval. Since movements bring to higher values, it is possible to classify those conditions with empirically determined threshold values, combined with a few sample delays to avoid spurious transitions.

In this way, we segment every movement that is encapsulated between two states when the device is still in user's hand. Since all the used gestures have limited duration, we added a condition on the minimum and maximum time for the movements to be segmented as gestures. This helps to avoid many unwanted movements being identified as gestures.

Despite those conditions, this algorithm still identifies a lot of random movements as gestures, leading to many false positive results from the classifier, as shown later in Section V. To improve the performance and the usability of the device, we introduced a new operation mode, called *Assisted Segmentation*. In this mode the user performs a shake gesture to enable and disable the segmentation and recognition of all the other gestures. The shake gesture is recognized using the segmentation FSM and has a high accuracy: during our tests we obtained a correct classification ratio of 100% within 80 executions of the gesture and only one false positive.

By default, gesture recognition is disabled, and the user can move the device in any way (e.g. walking with the device or using the device as a pointer on an interactive table). When needed, the user performs a shake to "wake up" the smart object, activating the recognition algorithm and then executes the wanted gestures to interact with the system. During this time the user can pay attention to the movements performed, to avoid the recognition of random movements as gestures. Another shake disables the interaction capabilities of the device, and the user can move it freely.

This user-assisted segmentation technique increased the overall performance of the device, reducing drastically the number of false positive recognitions.

B. HMM Gesture Recognition

The main feature used for gesture recognition is the direction of the movement, represented by the direction of the acceleration vector, sampled at each frame. This information is obtained converting the 3D acceleration vector $\{a_x, a_y, a_z\}$ in spherical coordinates, and using only the two angles $\{\varphi, \theta\}$.

To efficiently compute the two angles of the acceleration vector we implemented an algorithm based on the CORDIC algorithm [20]. Using the notation in Fig. 3, this algorithm first estimates the phase θ and the magnitude r' of the complex number $(a_x + ia_y)$, then again estimates the angle φ, using r' and a_z. All computations are done with integer values, giving us a resolution of 1 degree and a maximum error of 2 degrees, which is acceptable since we are dealing with human motions and don't need higher accuracy.

Discrete HMMs are less computationally demanding than those operating on continuous observations, so they are the best choice in our case, since we are focusing on a limited resource implementation. As input to the discrete models we need to use

 ISBN: 978-1-61208-100-7

a discrete feature symbol to represent the directional information. The two angles calculated, that identify the arbitrary 3D orientation of a unitary vector, are quantized to the nearest vector of the codebook by a simple minimum distance classifier. In this way, the stream of two angles is converted in a stream of codebook indices, which is a suitable input to discrete HMMs. The number of vectors in the codebook was empirically determined and a codebook with k = 26 vectors uniformly distributed on the spherical surface resulted to be the best trade-off between quantization accuracy and processing complexity.

The off-line HMM training phase builds a model for each of the gestures to recognize, using sample instances of the gestures. We used the Baum-Welch algorithm, and initialized the training models with several random probability distributions. Among the resulting HMMs, those with the lowest training error were chosen.

To improve the model behavior, when dealing with input gestures that are slightly different from those used during training, we modified the symbol observation probability distribution (i.e. the observation matrix B in the discrete case). A model with a uniform observation matrix B_0 recognizes every gesture with a same low probability. We interpolated the trained models with the uniform one, by weighting the observation matrixes with a factor ε as given by the equation

$$B' = \varepsilon B + (1 - \varepsilon) B_0. \quad (1)$$

The optimal ε factor was empirically obtained and lies in the range [0.7 – 0.9].

The on-line recognition algorithm evaluates the executed gesture with all of the trained models, and selects the model with the highest probability. For this purpose we used a fixed point version of the forward algorithm, as introduced in [22]. This implementation deals with the lack of a division unit in the low power microcontroller embedded in the device, and proposes a different scaling procedure that uses shifts and a logarithmic representation of the probabilities. Our previous work compared the performance of this implementation against a standard floating point algorithm. The results showed that a 16 bit fixed point algorithm has the best trade-off between classification rate and computational complexity.

V. ANALYSIS AND RESULTS

A. Experimental Setup

For the validation of our algorithm we used a set of 7 gestures, illustrated in Fig. 5. All gestures are formed by natural movements, start and end with the user holding the cube in the same position and are executed on the vertical plane in front of the user, holding the device every time in the same orientation.

We collected gestures executed by four people, all male students with an age of 26 years. To build and validate the HMMs each user executed 80 instances of every gesture, during different days. Those gestures were continuously executed, with a few seconds of interval between two consecutive instances, and segmented with our algorithm.

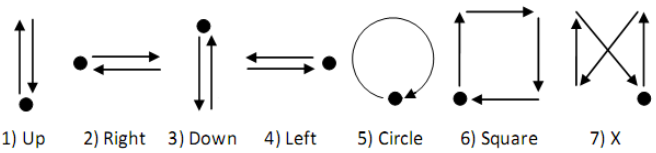

Figure 5. Gestures used for algorithm validation. The dots indicate the start and end position.

Gestures from three users were used for both modeling and validation, while gestures from the fourth user were used only to test the models. From each user we also collected several continuous streams, containing gestures and random movements, to simulate an actual usage of the device and test the overall recognition algorithm. The whole dataset was collected with the SMCube and sent via Bluetooth to a PC. No feedback from the device or the PC was given to the users during the execution of the gestures.

To easily test the performance, the algorithm was implemented in Matlab, taking care to simulate the computational constrains of the 8 bit microcontroller and using only integer computations with controlled variable size.

B. Test and Simulations

In the first place we used the collected dataset to train a set of HMMs for each user, using the floating point notation with double precision. Each model have been trained using 15 reference instances, 15 loops for the Baum-Welch training algorithm, and 10 initial random models. The floating point models were then converted in fixed point, represented only by 16 bit integers. Each user's models were validated with his/her own gestures, not used in the training phase, and with gestures from other users.

In Fig. 6 we present classification results in function of the interpolation factor ε. The circled points refer to the case when models trained by one user are validated with his own gestures; the triangular points when the models trained on a user are validated on the other user's gestures, and the squared points indicate when a global model is used on all users.

The classification performance is measured with the *Correct Classification Ratio* (CCR), defined as the ratio between the number of correctly classified instances and the total number of instances.

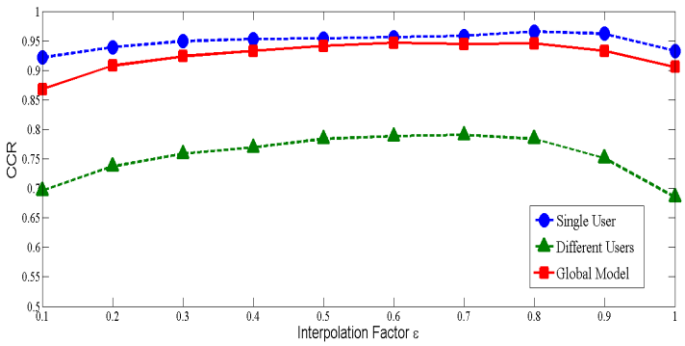

Figure 6. Average CCR for various ε values: the first line is in a single user scenario, the second is for the multi-user scenario and the third is for the global model

 ISBN: 978-1-61208-100-7

The results show how the classifier performs well in a single user scenario with recognition rates up to 99.7%. The algorithm has some limitations in a multi-user scenario, when recognizing gestures from a user with models trained by another user. We found that interpolating the trained models with a uniform one gave some advantages, and the best case is with an interpolation factor ε = 0.8. Table I shows the classification rates for the various users in this case; Tab. II shows the classification matrix in the best case and Tab. III in the worst case.

To overcome the limitations in the multi-user scenario, we put together all the gesture instances, regardless to the user who executed them, and build a global model for each gesture. These models were trained using 15 randomly chosen gestures and validated on 200 gestures from four users. The results of this case are presented in Fig. 6 with the squared points, where we can observe how in this case we have a performance comparable to the single user scenario, despite we are classifying gestures from all the users. We can also notice that the fixed point implementation has performances comparable to the floating point one, and our algorithm is suitable for low-performance smart objects. Table IV shows the correct classification matrix for the best case, with $\varepsilon = 0.6$.

With this global model we evaluated the overall algorithm performance, using the collected continuous streams of data which contain gestures and random movements. In this way we could test the segmentation and recognition algorithms together. Table V presents the results of this analysis. The automatic segmentation algorithm has good performance in recognition executed gestures, but gives also a lot of false positive results, identified by the insertions. The performance depends on what the user is doing and how the device is moved: long and continuous movements are easily rejected, but short movements, similar to gestures, trigger the recognition algorithm leading to a false positive. Deletions indicate how many times the algorithm misses a gesture, and this happens only if the gesture is executed too quickly or too slowly. Minimum and maximum duration times are derived from the collected dataset, and deletions may happen only in extremely short or long gestures.

To improve the device usability we proposed the assisted segmentation algorithm, which lets the user disable the recognition of gestures when not needed. Recognition rates for this algorithm are the same of the automatic one, since it uses the same HMMs, but in this case we have almost no insertions.

TABLE I. CLASSIFICATION RATES IN MULTI-USER SCENARIO

Training Set	Validation Set		
	User 1	*User 2*	*User 3*
User 1	99%	66.7%	85.9%
User 2	94%	92.8%	85.5%
User 3	93.3%	71.9%	99.7%

TABLE II. CCR BEST CASE: USER 3 USES HIS OWN MODEL

Performed gestures	Classified as						
	Up	*Right*	*Down*	*Left*	*Circle*	*Square*	*X*
Up	62	0	0	0	0	3	0
Right	0	65	0	0	0	0	0
Down	0	0	65	0	0	0	0
Left	0	0	0	65	0	0	0
Circle	0	0	0	0	65	0	0
Square	0	0	0	0	0	65	0
X	0	0	0	0	0	0	65

TABLE III. CCR WORST CASE: USER 2 USES MODEL TRAINED BY USER 3

Performed gestures	Classified as						
	Up	*Right*	*Down*	*Left*	*Circle*	*Square*	*X*
Up	33	0	0	0	1	28	3
Right	0	25	0	30	6	1	3
Down	12	1	43	1	0	7	1
Left	0	3	0	60	1	1	0
Circle	0	0	0	5	42	15	3
Square	0	0	0	0	13	50	2
X	3	0	15	3	2	3	39

TABLE IV. CCR FOR THE GLOBAL MODEL

Performed gestures	Classified as						
	Up	*Right*	*Down*	*Left*	*Circle*	*Square*	*X*
Up	194	0	3	0	2	1	0
Right	0	187	1	11	0	1	0
Down	1	0	199	0	0	0	0
Left	0	1	0	199	0	0	0
Circle	0	0	0	4	177	19	0
Square	0	0	0	0	13	187	0
X	3	0	12	0	1	2	182

TABLE V. GLOBAL MODEL CONTINUOUS RECOGNITION PERFORMANCES

	Auto Segmentation	*Assisted Segmentation*
Executed Gestures	83	78
Correctly Classified	71	62
Insertions	45	2
Deletions	6	5

C. Processing Perfomance Results

All the tasks needed for the gesture recognition algorithm were implemented on a PC in Matlab and on the ATmega168 microcontroller in C, using the AVR-GCC compiler and a 8 MHz clock. The Table VI presents the computational costs needed to perform the main operations at each frame. The Matlab implementation uses only fixed point operations on 16 bit integers, to simulate the embedded version and can be easily ported on other microcontrollers.

Each gesture model requires 3 matrices of 16 bit variables and with the implementation choices (7-state models with a 26 vector codebook) we need 462 Bytes to store each model. The microcontroller used has only 1 Kbyte of RAM, so we stored the models in the 16 KB of FLASH memory, used as program space. The entire application uses up to 12480 bytes of FLASH and 360 bytes of RAM memory.

We found a similar fixed-point implementation of HMMs in [2], which uses a 32-bit ARM7 microcontroller running at

 ISBN: 978-1-61208-100-7

65MHz. They recognize only 3 gestures and have recognition rates comparable to ours, but a shorter execution time (2.7 ms).

TABLE VI. COMPUTATIONAL COSTS

	ATmega168 (ms)	*PC-Matlab (ms)*
Preprocessing	0.03	0.37
Segmentation	0.20	0.40
Feature extraction	0.17	0.19
HMM (1 gesture)	0.73	0.83
HMM (7 gestures)	5.00	5.91
Total	6.13	7.70

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented an on-line segmentation and gesture recognition algorithm, implemented on a low-cost, low-power 8 bit microcontroller.

The automatic segmentation algorithm effectively finds executed gestures, but introduces also false positives. To address this issues, we introduced an *Assisted Segmentation* mode, which disables the gesture spotting algorithm when not needed, through the execution of a shake gesture.

We optimized a fixed point implementation of the HMM algorithm, which can be implemented on such low-performance microcontrollers, maintaining the same results as in a floating point case. The recognition algorithm can be used in a multi-user scenario, employing a global model trained with gestures from various users. It can be improved if the device is used only by one user, training the algorithm with only his/her own gestures.

In a future work we will explore the best ways to provide a feedback to the user, to see if the user can be trained to adapt his behavior and gestures in order to maximize the performance of the device.

VII. ACKNOWLEDGEMENT

Part of this work has been supported by SOFIA project funded under the European Artemis programme SP3 Smart environments and scalable digital service (Grant agreement: 100017) (www.sofia-project.eu).

REFERENCES

[1] O. Amft, H. Junker, and G. Troster, "*Detection of eating and drinking arm gestures using inertial body-worn sensors*", in Proceedings of the 9th IEEE international Symposium on Wearable Computer (ISWC), pp.160-163, 2005.

[2] R. Amstutz, O. Amft, B. French, A. Smailagic, D. Siewiorek, and G. Troster, "*Performance Analysis of an HMM-Based Gesture Recognition Using a Wristwatch Device*", In Proceedings of the 2009 international Conference on Computational Science and Engineering – Vol. 02, pp.303-309, 2009.

[3] G. Bailador, D. Roggen, G. Tröster, and G. Trivino, "*Real time gesture recognition using continuous time recurrent neural networks*", in 2nd Int. Conf. on Body Area Networks (BodyNets), Article n°15, 2007.

[4] S.Baraldi, L.Benini, O.Cafini, A.Del Bimbo, E.Farella, L.Landucci, A.Pieracci, and N.Torpei, "*Introducing TANGerINE: A Tangible Interactive Natural Environment*", in proceedings of ACM MultiMedia 2007, pp.831-834, 2007

[5] M. Bruns Alonso, and V. Keyson, "*MusicCube: a physical experience with digital music*", Personal Ubiquitous Comput., Vol.10, Issue 2-3, pp.163-165, 2006.

[6] A. Cayci, J. B. Gomes, A. Zanda, E. Menasalvas and S. Eibe, "*Situation-Aware Data Mining Service for Ubiquitous Environments*", Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), pp.135-140, 2009.

[7] G. S. Chambers, S. Venkatesh, G. A. West, and H. H. Bui, "*Segmentation of Intentional Human Gestures for Sports Video Annotation*", in Proceedings of the 10th international Multimedia Modelling Conference, pp.124-130, 2004.

[8] F. G. Hofmann, P. Heyer, and G. Hommel, "*Velocity Profile Based Recognition of Dynamic Gestures with Discrete Hidden Markov Models*". in Proceedings of the international Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction, pp.81-95, 1997.

[9] C.-R. Huang, C.-S. Chen, and P.-C. Chung, "*Tangible photorealistic virtual museum*", IEEE Comput. Graph. Appl., Vol.25(1), pp. 15-17, 2005.

[10] S. Jordà, M. Kaltenbrunner, G. Geiger, and M. Alonso, "*The reacTable: a tangible tabletop musical instrument and collaborative workbench*", in Proceedings of the International Conference on Computer Graphics and Interactive Techniques, Article n°91, 2006.

[11] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca, "*Accelerometer-based gesture control for a design environment*". Personal Ubiquitous Comput., Vol.10 (5), pp. 285-299, 2006.

[12] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "*An online algorithm for segmenting time series*", in Proceedings of hte 2001 IEEE International Conference on Data Mining, pp.289-296, 2001.

[13] L. Kim, H. Cho, S. H. Park, and M. Han, "*A tangible user interface with multimodal feedback*", in Proceedings of the 12th international conference on Human-computer interaction, pp. 94-103, 2007.

[14] C. Lee and Y. Xu, "*Online, Interactive Learning of Gestures for Human/Robot Interfaces*", 1996 IEEE International Conference on Robotics and Automation, pp. 2982-2987, 1996.

[15] V.-M. Mantyla, J. Mantyjarvi, T. Seppanen, and E. Tuulari, "*Hand gesture recognition of a mobile device user*", IEEE International Conference on MultiMedia and Expo, Vol: 1 (c), pp. 281-284, 2000.

[16] C. O'Malley and D. Stanton Fraser, "*Literature Review in Learning with Tangible Technologies*", Technical report, url: http://www.telearn.org/open-archive/browse?resource=298_v1, last access on 25/05/2010.

[17] L. R. Rabiner, "*A tutorial on hidden Markov models and selected applications in speech recognition*", in Proceedings of the IEEE, Vol.77(2), pp.267-296, 1989.

[18] B. Schiettecatte and J. Vanderdonckt, "*AudioCubes: a distributed cube tangible interface based on interaction range for sound design*", in Proceedings of the 2nd international Conference on Tangible and Embedded interaction (TEI), pp.3-10, 2008.

[19] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowicz, and G. Tröster, "*Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario"*, 10th IEEE International Symposium on Wearable Computers (ISWC), pp. 97-104, 2006.

[20] J. Volder, "*The CORDIC computing technique*", IRE Trans. Electron. Comput., pp. 257-261, 1959.

[21] J. A. Ward, P. Lukowicz, G. Troster, and T. E. Starner, "*Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers*". IEEE Trans. Pattern Anal. Mach. Intell. Vol.28, pp. 1553-1567, 2006.

[22] P. Zappi, B. Milosevic, E. Farella and L. Benini, "*Hidden Markov Model based gesture recognition on low-cost, low-power Tangible User Interfaces*", Entertainment Computing, Volume 1 (2), 75-84, 2009.

 ISBN: 978-1-61208-100-7