Situation Awareness in a Smart Home Environment

Shu-Yun Lee and Fuchun Joseph Lin
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
{ happy8155.cs03g | fjlin}@g2.nctu.edu.tw

***Abstract* — Situation awareness is a must for a smart home to exhibit its smartness. Normally, this is accomplished by accurately detecting the activities of a home user and then responding to the need of the user accordingly. This research utilizes a single wearable device equipped with an accelerometer and a gyroscope to detect eight potential activities in the living room of a smart home environment. First, the models of activities are constructed based on training data generated from the wearable device. Then, when a user performs the activity, the newly generated data would be compared with the established models to identify the type of current activity. Our method of model construction and activity detection is based on Decision Tree and Hidden Markov Model (HMM) with the assistance of location data derived from Beacons. The unique advantage of our method lies in its low cost as only one wearable device and a couple of beacons are required for achieving the desired situation awareness.**

Keywords—Wearable Device; Internet of Things; Decision Tree; Hidden Markov Model

I. INTRODUCTION

With proliferation of IoT technologies, smart devices are now closely connected to our life. Many of these devices such as smart phones, wearable devices and tablets greatly improve the quality of our life. In particular, wearable devices have become more and more popular in recent years [1]. There are diverse areas of applications developed for wearable devices [2] such as communications, entertainment, sport and lifestyle. Among these, lifestyle is considered as the dearest to our daily life, especially in a smart home environment.

This research utilizes a single wearable device and three Beacons to identify a user's eight daily activities in the living room of a smart home environment, including watching TV, reading newspaper, chatting with other family members, lying down on the sofa for a nap, listening to music, doing yoga, enjoying massage or just walking around. The user is only required to wear a wearable device on the right waist (Fig. 1) with three Beacons installed in the living room. Assuming that the user would carry his/her own BLE-equipped cellphone all the time, the unique advantage of our solution is that it requires neither multiple wearable devices nor any camera to detect a user's activity. Through its situation awareness, our system is capable of reacting to the user's need with appropriate environment adjustment in terms of light and music control.

The rest of the paper is organized as follows: Section II surveys related work. Section III introduces the background including both hardware and software components required by our system. Section IV explains our system architecture and its three design alternatives. Section V describes the algorithms for activity learning and activity identification. Section VI presents our experimental results and compares three design alternatives. Finally, in Section VII we provide our conclusion and future work.

II. RELATED WORK

We survey related work in activity recognition using wearable devices. Most methods use more than one wearable device but produce less accurate results than our approach. Due to the space limit, only two of them are presented below.

A. *Wearable Device and Camera-Based Activity Recognition*

Chun et al. [3] proposed a two-step recognition algorithm for indoor daily activity identification using both motion data and location data: (1) coarse-grained classification (2) fine-grained classification. The coarse-grained classification combines the outputs of two neural networks to classify activities. The fine-grained classification applies a modified short-time Viterbi algorithm to solve the hidden states of Hidden Markov Model (HMM) and classifies activities in further detail. Then, they combine the motion data and location information to reach the final result.

B. *Multiple Wearable Sensor-Based Real-Time Activity Recognition*

Liang et al. [4] proposed a hierarchical approach for real-time activity recognition with multiple sensors attached to a user or embedded in the environment. A high-level activity usually includes gestures and ambulation. To recognize the activities hierarchically, they first identify gestures at the sensor node level then recognize complex activities at the portable device level. To identify gestures, they use a K-Medoids clustering method to discover template gestures and Dynamic Time Wrapping (DTW) to match the templates using Euclidean distance. After matching, they use the Emerging Pattern based algorithm to recognize the activities in real time. They did experiments for 26 activities and most of them were recognized accurately. However, few of them such as making coffee were recognized inaccurately (4.3%). The variance of the accuracies is very large.

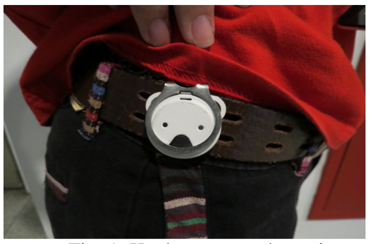


Fig. 1. Koala worn on the waist

C. *Summary*

To identify activities, some research would use a single wearable device and location information while the other would use multiple sensors. They usually use Hidden Markov Model (HMM) and Viterbi algorithm to find the hidden states. In addition to related work we mentioned above, we also survey other research work in activity identification [5] [6]. The overall accuracy is above 70%. Most of them are about 85%.

III. BACKGROUD

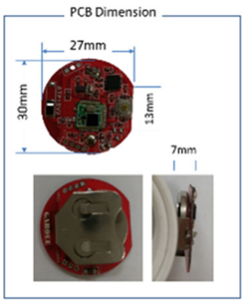
We explain sensors, processing hardware and mining tools required by our system in the following paragraphs.

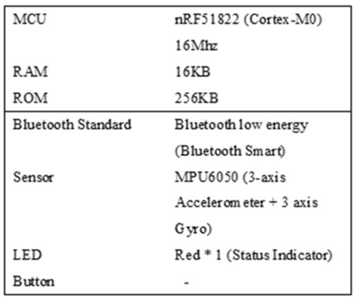
A. *Sensors*

Both motion data and location data are required by our system to identify the user's activity. The primary device used for collecting both motion data and location data is a wearable device called Koala as shown in Fig. 1.

1) *Wearable Device*

Koala is developed by National Chiao Tung University. It contains a 3-axis accelerometer and a 3-axis gyroscope, and uses Bluetooth Low Energy (BLE) as the communication protocol. Its specification is shown in Fig. 2. Any BLE-equipped smart phone can establish connection with Koala and receive approximately 30 sets of acceleration and gyroscope data per second. Due to the frequency of motion data, the window size is set at 60 data and the movement is set at every 15 data. When the window is full, the system will execute algorithms to identify the user's current situation, then move the window to collect next 15 motion data.





MCU	nRF51822 (Cortex-M0) 16Mhz
RAM	16KB
ROM	256KB
Bluetooth Standard	Bluetooth low energy (Bluetooth Smart)
Sensor	MPU6050 (3-axis Accelerometer + 3 axis Gyro)
LED	Red * 1 (Status Indicator)
Button	-

Fig. 2. Specification of Koala

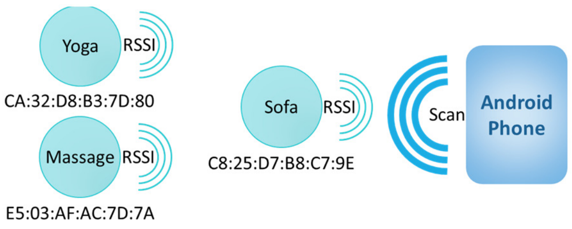


Fig. 3. Scanning Beacons with an Android smart phone

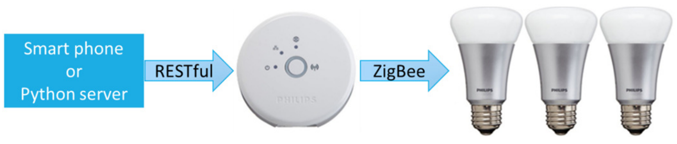


Fig. 4. Hue control flow

2) *Beacon*

To retrieve the user's location data, the system uses three Beacons located at three locations in the living room. The Beacon broadcasts messages periodically and the message contains MAC (Media Access Control) address and RSSI (Received Signal Strength Indicator) value. With the MAC address, the system can know the Beacon's location via the Beacon's identity as shown in Fig. 3. With the RSSI value, the system can further decide which Beacon is the closest one to identify the user's current location.

B. *Processing Hardware*

To provide a comfortable environment for the user in a home, the system would send control signal to the smart light and the music player to change the environment. We use Philips Hue [5] to be our smart light. Our system sends control signal to a Hue bridge through RESTful (REST is Representational State Transfer) communications. A Hue bridge is connected to the Ethernet and it can process control signals to control Hue bulbs separately or in group through ZigBee as shown in Fig. 4. A Hue bulb can be changed with different brightness and 16 million colors. With different light and music, the user can enjoy a more comfortable home environment.

C. *Mining Tool*

To build the Decision Tree for activity identification, we use Weka as our mining tool. Weka is the abbreviation of Waikato Environment for Knowledge Analysis [6]. It supports many machine learning algorithms including Decision Tree. To build the model, we collect training data first, calculate their features (mean, standard deviation and range) and save these features as a CSV file to be processed by Weka. Then, Weka would generate the Decision Tree model for activity identification.

IV. SYSTEM ARCHITECTURE

We explain our system architecture and its three design alternatives in this section.

Two types of devices are used in our architecture: Koala wearable device and Beacon. The former is used to collect the user's motion data while the latter is used for detecting the user's location. In addition, a smart phone is utilized to receive the data from both the Koala and the Beacons. We propose three architecture alternatives to process the raw data from the Koala and the Beacons for the purpose of identifying a user's activity.

The three architecture alternatives are depicted in Figs. 5, 6 and 7, respectively. Their difference lies in where the situation detection procedures will be carried out. The first architecture in Fig. 5 carries out all the situation detection procedures on the smart phone including both feature extraction and situation detection. The smart phone will then play music on itself and also send control signals to the Hue to provide the user with a comfortable environment.

On the other hand, the second and the third architectures in Figs. 6 and 7 use a backend Python server for data processing. The Python server is running on a Raspberry Pi 2 based on Python 2.7.9. The difference between two architectures lies on where the feature calculation is performed. The second architecture would send all the raw data to the backend Python server for processing (Fig. 6) while the third architecture would calculate the features on the smart phone but send the features to the Python server for further processing (Fig. 7).

Then, based on the user's specific activities different signals would be sent by Python Server to control the music player on the smart phone and the Hue in the smart home environment. We're to find out which architecture alternative is the best among the three.

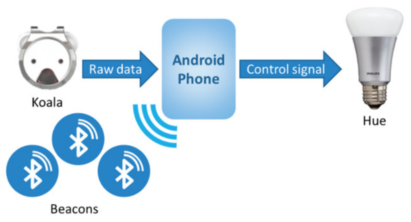


Fig. 5. First architecture with all processing on the smart phone

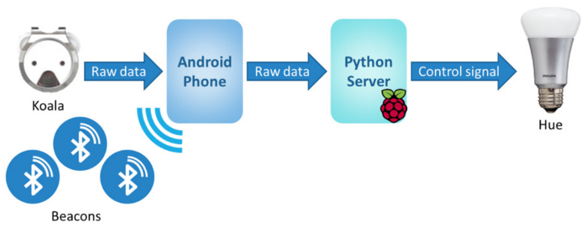


Fig. 6. Second architecture with all processing on the Python server

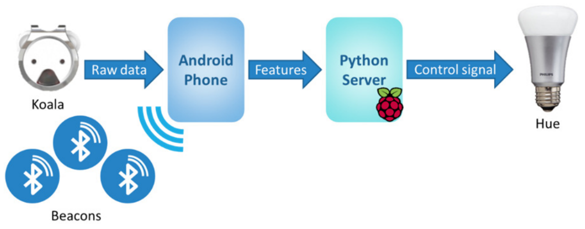


Fig. 7. Third architecture with processing distributed on a smart phone and a Python server

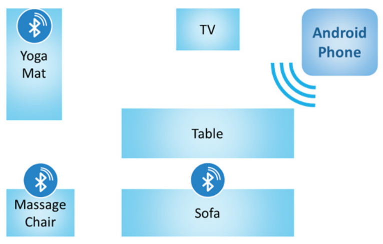


Fig. 8. Beacons distribution

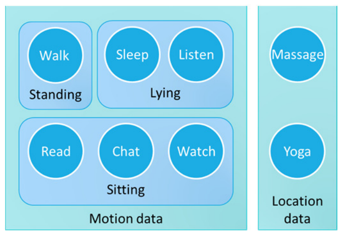


Fig. 9. Classifications of activities

V. Activity Detection Algorithms

We describe the algorithms for activity learning and activity identification in this section.

A. *Location Identification*

Location identification is accomplished by Beacons. With the smart phone, Beacons stuck on the sofa, yoga mat and massage chair by the tape (Fig. 8) can be scanned and identified based on their MAC addresses. Also, with the RSSI values from the Beacons, the distance can be measured between each Beacon and the smart phone.

The smart phone keeps scanning Beacons and receiving RSSI values. We compare the RSSI values from three Beacons. The maximum RSSI value indicates where the user is located. On the other hand, if all RSSI values are smaller than or equal to -65, this indicates that the user is nowhere near sofa, yoga mat or massage chair but walking around.

B. *Activity Identification*

Both motion data and location data (Fig. 9) are required to identify a user's activity. For motion data, both Decision Tree and Hidden Markov Model (HMM) algorithms [7] are used for activity analysis and identification. Before applying these algorithms, a mining tool and several mathematical methods are used to construct the models.

1) *Decision Tree*

We collect training data for each activity and use Excel to calculate the features of raw data. The features calculated includes mean, standard deviation and range. The raw data contains 3-axis acceleration, 3-axis gyroscope and total acceleration data. Therefore, we would generate 21 features.

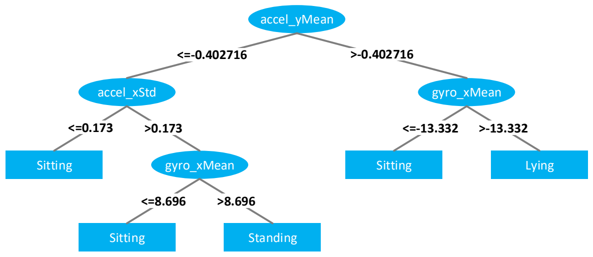


Fig. 10. Decision Tree for sitting, lying and standing

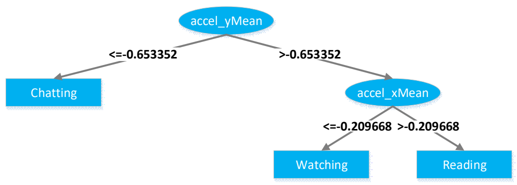


Fig. 11. Decision Tree for chatting, watching and reading

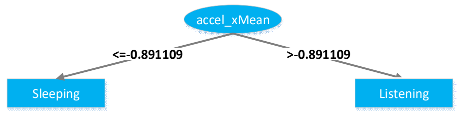


Fig. 12. Decision Tree for sleeping and listening

After features generation, Weka is employed to build a Decision Tree model (as illustrated in Fig. 10) to identify three classifications of activities including sitting, lying and standing which cover 6 activities on the left of Fig. 9.

After the first step, we can identify walking as it is the only one classified as standing. We then use the Decision Tree in Fig. 11 to further classify sitting into chatting, watching and reading. In the last step, the Decision Tree in Fig. 12 is used to further classify lying into sleeping and listening. The activities of enjoying massage and doing yoga can be determined just by location data. As a result, we can classify all activities.

When the raw data is coming in, the application can calculate the features every 60 data with the window's movement at every 15 data. Then, the application that implements the Decision Tree of Figs. 10-12 can identify the user's activities.

After this procedure, we determine the user's location by the data from Beacons. If the user is on the massage chair, we recognize the activity as enjoying massage. If the user is on the yoga mat, we classify the activity as doing yoga. If the user is far from Beacons, it means the user is walking around. If the user is on the sofa, it means the activity is one of five we would identify by Decision Tree.

Though we can identify all the user's activities by the above procedures, the accuracies of some activities identification are lower than 90% if Decision Tree is used alone for identification. Table 1 shows that when Decision Tree is used alone to identify six activities (watching, reading, chatting, sleeping, listening and walking), the average accuracy of lying is higher than 90% but the accuracies of sitting and standing are lower than 90%.

For the low accuracy of standing identification in Table 1, we use location data to improve its identification accuracy. Then for the low accuracy of sitting identification in Table 1, we use Hidden Markov Model (HMM) to improve its identification accuracy.

2) *Hidden Markov Model*

Hidden Markov Model (HMM) [7] uses the observable states to inference the hidden states. We use Decision Tree to perceive the observable states and Hidden Markov Model to inference the hidden states and generate the identification result.

HMM requires three probabilities: initial probability, observation probability and transition probability. Before a user starts his/her activities in a living room, he/she has to walk into the living room first. Therefore, the initial probability is set as Table 2.

Table 1. Accuracy of using Decision Tree only

Classification	Sitting			Lying		Standing
Activity	Read	Chat	Watch	Listen	Sleep	Walk
Accuracy	83%	94%	88%	93%	90%	70%

Table 2. Initial probability matrix

Activity	Watch	Read	Chat	Sleep
Probability	0	0	0	0
Activity	Listen	Walk	Yoga	Massage
Probability	0	1	0	0

Table 3. Observation probability matrix

State \Observation	Watch	Read	Chat	Sleep	Listen	Walk	Yoga	Massage
Watch	0.83	0	0.12	0	0	0.05	0	0
Read	0	0.78	0.17	0	0	0.05	0	0
Chat	0	0.06	0.89	0	0	0.05	0	0
Sleep	0	0	0	1	0	0	0	0
Listen	0	0	0	0	1	0	0	0
Walk	0	0	0	0	0	1	0	0
Yoga	0	0	0	0	0	0	1	0
Massage	0	0	0	0	0	0	0	1

Table 4. Transition probability matrix

Before \ After	Watch	Read	Chat	Sleep	Listen	Walk	Yoga	Massage
Watch	0.88	0.06	0.03	0.01	0.01	0.01	0	0
Read	0.09	0.82	0.06	0.01	0.01	0.01	0	0
Chat	0.1	0.2	0.67	0.01	0.01	0.01	0	0
Sleep	0.2	0.2	0.2	0.2	0.2	0	0	0
Listen	0.2	0.2	0.2	0.2	0.2	0	0	0
Walk	0.18	0.18	0.2	0	0	0.4	0.02	0.02
Yoga	0	0	0	0	0	0.2	0.8	0
Massage	0	0	0	0	0	0.2	0	0.8

$$V_i = \begin{cases} P(y|i) * \pi_i, if\ it\acute{}s\ the\ first\ time\ to\ calculate\ V_i \\ max_{x \in S}\left(P(y|i) * a_{x,i} * V_x\right), else \end{cases}, i \in S$$

Formula 1. Probability of each state

$$\mathrm{x} = \mathrm{argmax}_{i \in S}(Vi)$$

Formula 2. Formula of finding the most probable state

Observation probability (Table 3) *P(y|i)* means the probability of Current State = *i* and Observation State = *y*. Transition probability (Table 4) means the probability of transition from the original state to the next state. Both the observation probability matrix and the transition probability matrix are generated by the experimental results using Decision Tree. With HMM, we can find the most probable state. To solve the HMM, we need to use the Viterbi algorithm [9].

3) *Viterbi Algorithm*

The Viterbi algorithm is used to solve our Hidden Markov Model (HMM). The Viterbi algorithm is a dynamic programming algorithm for finding the most probable sequence in HMM. Since we only need to know a current state, we modify the original Viterbi algorithm to suit our purpose.

Via Formula 1, we can calculate the probability of each state.

i belongs to set *S*, the set of activities.

P(y|i) means the probability of Current State = *i* and Observation State = *y*.

π_i is the probability of Initial State = *i*.

$a_{x,i}$ signifies the transition probability from State *x* to State *i*.

If it's the first time to calculate Vi, we use the upper part of Formula 1. If it's not the first time, we use the lower part of Formula 1. *Vx* in the lower part means the result from last time.

After calculating the probability of each state, we use Formula 2 to find the most probable state. Formula 2 signifies V_x is the maximum probability among all V_i where i means each activity. As a result, we can identify the user's current activity.

C. *Smart Appliance Control*

After all model construction and activity recognition, we can know the user's current activity. In response to different activities, different music and Hue control as indicated in Table 5 will be provided. For example, a moderate light will be provided for watching TV. For reading newspaper, the user will be given brighter illumination. When the user is chatting with other family members, we change Hue to a warm mood. If we detect the user is lying down on the sofa to take a nap, we dim the light. We provide a colorful environment to make the user enjoy listening to music. When the user is walking around, we just make the light on. If the user is doing yoga, we make him/her feel like in a forest to relax his/her mind. If the user is enjoying massage, we play the soft music and a relaxing light. For a different situation, the system would respond with a different yet the most comfortable environment for the user.

VI. Experimental Results

We test whether using Hidden Markov Model (HMM) and Viterbi algorithm can reach a higher accuracy than only using Decision Tree. We also compare the performance of three design alternatives discussed in Section 3. This section shows our experimental results.

A. *Accuracy Comparison between with and without Viterbi Algorithm under different architectures*

To compare the accuracy between with and without Viterbi algorithm, we use three different processing architectures.

Architecture 1.	All processing done on the smart phone
Architecture 2.	All processing done on a Python server
Architecture 3.	Processing distributed on both the smart phone and the Python server

Fig. 13 shows the result of using Architecture 1. Most activity identification can reach a higher accuracy with Viterbi algorithm. The only exception is sleeping (79.22%) when comparing to those not using Viterbi algorithm (84.11%). This is because during sleeping the Koala's position may be changed to interfere our detection algorithm. However in average, we can still reach a higher activity identification accuracy (92.13%) using Viterbi algorithm than not using it (83.62%).

Table 5. Mapping of activities and a smart home environment

	Watch	Read	Chat	Sleep
Music	X	X	X	X
Color	🟨	⬜	🟧	🟫

	Listen	Walk	Yoga	Massage
Music	X	X	relaxing	love
Color	🟪	🟧	🟩	🟪

In Architecture 2, the smart phone would send to the Python server (1) 15 sets of raw data from the Koala and (2) the user's location data scanning from Beacons. The Python server processes those data to identify the user's activity. The accuracy results are shown in Fig. 14. Using Viterbi algorithm can reach a higher accuracy for five of the cases. However, the average accuracy of using Viterbi algorithm (90.99%) is lower than only using Decision Tree (94.79%). This is likely caused by the unstable position of the Koala on user's waist.

In Architecture 3, the smart phone calculates the features first and then sends the features and location data to the Python server for processing. The result is shown in Fig. 15. A half of the cases reaches higher accuracies with Viterbi algorithm than without Viterbi algorithm. Another half of them using Viterbi algorithm are less accurate than not using it (94.40% vs. 94.57%, 95.05% vs. 100%, 95.35% vs. 98.95% and 92.59% vs. 96.58%).

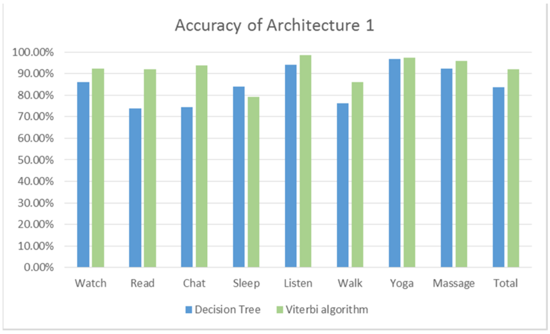


Fig. 13. Accuracy of Architecture 1

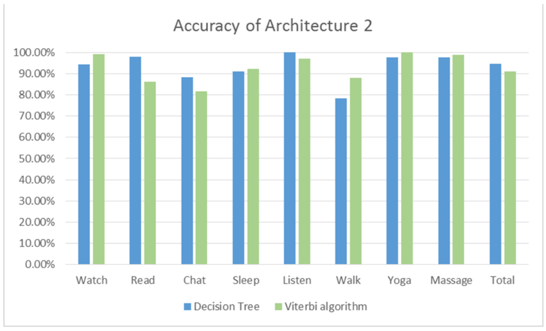


Fig. 14. Accuracy of Architecture 2

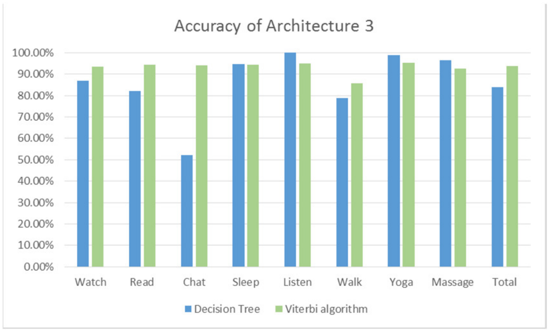


Fig. 15. Accuracy of Architecture 3

Table 6. Experimental accuracies

	Viterbi algorithm	**Decision Tree**
Android only	92.13%	83.62%
Android + server (Raw data)	90.99%	94.79%
Android + server (Feature)	93.81%	83.83%
Average	92.31%	87.41%

Table 7. Time cost of different cases (Unit: second)

	Architecture 1	**Architecture 2**	**Architecture 3**	**Average**
Viterbi algorithm	0.636	0.575 (0.564+0.011)	0.557 (0.554+0.003)	**0.589**
Decision Tree	0.588	0.608 (0.552+0.056)	0.578 (0.576+0.002)	0.59
Average	0.612	0.5915	**0.5675**	

But the differences are very small (0.17%, 4.95%, 3.6% and 3.99%), so the results are still acceptable. Overall, the average accuracy (93.81%) of using Viterbi algorithm is still higher than using Decision Tree only (83.83%). Table 6 summarizes our experimental results of three architectures. We can see the average accuracy of using Viterbi algorithm is higher than only using Decision Tree. We thus can conclude that using Hidden Markov Model (HMM) and Viterbi algorithm are more suitable for activity identification than only using Decision Tree. The overall results show that the architecture plays little role in the accuracies of the results.

B. *Performance comparison under different architctures*

Due to transmission time difference caused by different network environment, only computational time on different architectures is used for performance comparison. Our experimental results are shown in Table 7. The first value in parentheses is the computational time on a smart phone and the second one is the computational time on the Python server. The value before parentheses is the sum of both.

In the case of using Viterbi Algorithm, Architecture 3 where processing is distributed on a smart phone and a server is able to reach the highest efficiency (0.557 sec) than the other two (0.636 sec and 0.575 sec). This is because a Python server always performs better than a smart phone. In addition, transmitting features is always more efficient than transmitting large raw data.

In the case of using Decision Tree, Architecture 3 also has the best performance (0.578 sec) than the other two (0.588 second 0.608 sec). We infer the same reason as that in the case of Viterbi Algorithm.

Though Architecture 3 is able to reach the best average performance (0.5675 sec) among all three architectures, this comparison doesn't take the transmission time into consideration. As Architectures 2 and 3 always involve a server with potentially very long transmission time, it is possible that in the real environment Architecture 1 may still end up to be the most efficient design.

If we compare the performance between using Viterbi algorithm and Decision Tree, Table 7 shows that the average performance of using Viterbi algorithm is better in Architectures 2 and 3 but worse than in Architecture 1. This is likely caused by the fact that it is less efficient to perform Viterbi algorithm on a smart phone than on a server.

VII. Conclusions and Future Work

In this research, we use the user's motion data from a Koala wearable device and the location data from three Beacons to identify the user's situation in a smart home environment: watching TV, reading newspaper, chatting with other family members, lying down on the sofa for a nap, listening to music, doing yoga, enjoying massage and walking around. For a different situation, the system would respond with a distinct environment to the user by controlling the music player and smart lighting.

Between HMM/Viterbi algorithm and Decision Tree algorithm, we conclude through our experimental result that using HMM and Viterbi algorithm would be more accurate and have better performance than using Decision Tree alone. Also, among three different architectures of implementation, when taking transmission time into consideration, Architecture 1 where all processing is done on the smart phone is likely to be the best choice among all three alternatives.

Via identifying a user's activity, our smart home system can create a suitable and comfortable environment for the user through controlling the music player and smart lighting. Several areas still exist for future research such as developing multi-person activity identification, expanding the scope of service area and controlling more smart appliances in the home.

Acknowledgment

The project reported in this paper is sponsored by Ministry of Science and Technology (MOST) of Taiwan Government under Project Number MOST 104-3115-E-009-002.

References

[1] G. Gross, "The wearable market saw a big boom in '15," 23 02 2016. [Online]. Available: http://www.computerworld.com/article/3036756/wearables/the-wearable-market-saw-a-big-boom-in-15.html. [Accessed 05 2016].

[2] S. Romeo, "MORPHING TECHNOLOGY AND AESTHETICS FOR THE FUTURE OF WEARABLE DEVICES," 27 02 2014. [Online]. Available: http://www.telit2market.com/3392/morphing-technology-aesthetics-future-wearable-devices/. [Accessed 05 2016].

[3] C. Zhu, W. Sheng, "Motion- and location-based online human daily activity recognition," *Pervasive and Mobile Computing,* pp. 256-269, 2010.

[4] L. Wang, T. Gu, X. Tao, J. Lu, "A hierarchical approach to real-time activity recognition in body sensor networks," *Pervasive and Mobile Computing,* pp. 115-130, 2010.

[5] C. Zhu and W. Sheng, "Realtime Recognition of Complex Human Daily Activities Using Human Motion and Location Data," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING,* pp. 2422-2430, 9 2012.

[6] S. Forsström and V. Kardeby, "Estimating Contextual Situations using Indicators from Smartphone Sensor Values," *2014 IEEE International Conference on Internet of Things,* pp. 243-250, 2014.

[7] "Philips Hue," Philips, [Online]. Available: http://www2.meethue.com/. [Accessed 6 2016].

[8] "Weka (machine learning)," 04 2016. [Online]. Available: https://en.wikipedia.org/wiki/Weka_(machine_learning). [Accessed 05 2016].

[9] "Markov Model," [Online]. Available: http://www.csie.ntnu.edu.tw/~u91029/HiddenMarkovModel.html. [Accessed 4 2016].

[10] "Viterbi algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Viterbi_algorithm. [Accessed 15 4 2016].

[11] I. Witten, E. Frank, M. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Elsevier Science Ltd, 2011.