SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2017, Vol. 60 072104:1–072104:15
doi: 10.1007/s11432-015-1014-7

Personalized gesture interactions for cyber-physical smart-home environments

Yihua LOU[1], Wenjun WU[1*], Radu-Daniel VATAVU[2] & Wei-Tek TSAI[1]

[1]*State Key Laboratory of Software Development Environment, Beihang University, Beijing* 100191*, China;*
[2]*MintViz Research Lab, Integrated Center for Research, Development, and Innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for Fabrication and Control (MANSiD) & Department of Computer Science, University Stefan cel Mare of Suceava, Suceava* 720229*, Romania*

Received March 16, 2016; accepted April 22, 2016; published online October 13, 2016

Abstract A gesture-based interaction system for smart homes is a part of a complex cyber-physical environment, for which researchers and developers need to address major challenges in providing personalized gesture interactions. However, current research efforts have not tackled the problem of personalized gesture recognition that often involves user identification. To address this problem, we propose in this work a new event-driven service-oriented framework called gesture services for cyber-physical environments (GS-CPE) that extends the architecture of our previous work gesture profile for web services (GPWS). To provide user identification functionality, GS-CPE introduces a two-phase cascading gesture password recognition algorithm for gesture-based user identification using a two-phase cascading classifier with the hidden Markov model and the Golden Section Search, which achieves an accuracy rate of 96.2% with a small training dataset. To support personalized gesture interaction, an enhanced version of the Dynamic Time Warping algorithm with multiple gestural input sources and dynamic template adaptation support is implemented. Our experimental results demonstrate the performance of the algorithm can achieve an average accuracy rate of 98.5% in practical scenarios. Comparison results reveal that GS-CPE has faster response time and higher accuracy rate than other gesture interaction systems designed for smart-home environments.

Keywords gesture interaction, personalization, gesture recognition, user identification, event-driven architecture

Citation Lou Y H, Wu W J, Vatavu R D, et al. Personalized gesture interactions for cyber-physical smart-home environments. Sci China Inf Sci, 2017, 60(7): 072104, doi: 10.1007/s11432-015-1014-7

1 Introduction

Smart homes are intelligent environments where both physical home appliances and computing devices are interconnected to deliver easy-to-access, convenient, and highly interactive experiences for residents. For such hybrid cyber-physical smart-home environments, traditional input devices for human-computer interaction, such as the keyboard and the mouse, are no longer suitable. Although remote controls still represent the industry standard for interacting with household electrical appliances, these standard interfaces are sometimes difficult to control by users [1] and inappropriate for some tasks and contexts [2]. As a suitable alternative, gestural interfaces can deliver natural and intuitive interactions to their users [3,4]. Despite many efforts on integrating gestural interfaces for smart-home environments [5–7], the following challenges still hamper wider adoption of gestural interfaces:

* Corresponding author (email: wwj@nlsde.buaa.edu.cn)

1. Complexity of gesture recognition algorithms for system developers, especially for those who develop systems for new environments with none or few software resources (e.g., gesture libraries) available.

2. Complexity of gesture training procedures for end users. It is a non-trivial task for average users without technical backgrounds to train accurate gesture classifiers, and smart-home interfaces need to provide transparent ways to reduce the need for and implications of complex training procedures.

3. Complexity of identification of personalized gesture commands. When performing gestures, users may produce significant variations. Consequently, gesture recognition algorithms need to process user-specific personalized gesture preference profiles to achieve accurate classification. It is also necessary to have user identification mechanisms to make sure each user can be associated with his gesture preference profile.

To address the first challenge, using service-oriented approaches is a promising option [8, 9], as it hides the complex details of implementation while exposing clear ready-to-use services for the application designers of smart-home interaction systems. Due to the fact that human gestures are naturally event-driven, our previous work gesture profile for web services (GPWS) [9] has provided a good start point in implementing service-oriented framework for gestural interfaces using stateless services and the lightweight event-driven architecture. However, the other two challenges about the complexity of personalized gesture training and recognition remain unsolved. In this paper, we extend our previous work on GPWS and propose a new service-based personalized gesture interaction framework for smart homes called gesture services for cyber-physical environments (GS-CPE), which incorporates user identification and personalized gesture recognition through an event-driven architecture to solve all three challenges. Our new framework enables the residents of a smart home to control appliances in a natural and intuitive way due to its new features. The major contributions of this paper include as follows:

1. We introduce GS-CPE as an integrated framework for supporting personalized gesture interaction in cyber-physical smart-home environments. Previous gestural interfaces have not considered user identification, which means such systems can neither be optimized for personalized interaction nor be deployed in scenarios where user identity must be verified. GS-CPE addresses these aspects by integrating the user identification service that identifies users through gesture passwords to associate a user with the pre-stored gesture preference profile and the personalized gesture recognition service that recognizes personalized gestures through the gesture preference profile for an identified user. Through the service-oriented and event-driven architecture, GS-CPE provides easy-to-use APIs for system developers to implement a full-functional gesture interaction system.

2. We introduce a new gesture password recognition algorithm that adopts a two-phase cascading classifier for user identification. In GS-CPE, user identification works by recognizing human gesture motions as passwords consisting of symbols from a fixed user-independent symbol gesture set. Most gesture systems employ statistical-model classifiers to handle gesture password recognition, which often requires a large training dataset to ensure accurate performance. It is difficult for such a classifier to achieve a high recognition accuracy through a relatively small training set in smart-home environments. Therefore, we introduce a new cascading classifier by applying a template-matching classifier that requires much fewer training samples to achieve a high recognition accuracy as the second-level classifier, so that the overall recognition accuracy can be improved with less training.

3. We introduce a new personalized gesture recognition algorithm that supports multiple gestural input sources and dynamic template adaptation. Previous gesture interaction frameworks mostly rely upon a single input source to recognize human gestures, which often reject unreliable results merely based on likelihood values. However, as the range of likelihood values varies in different situations, it is difficult to determine an appropriate threshold for the rejection likelihood. To overcome this problem, we introduce a new personalized gesture recognition algorithm that employs multiple input sources to remedy possible bias caused by a single source. Instead of using a rejection likelihood threshold, our algorithm makes the rejection decision by comparing the classification results from different sources. Also, to overcome the problem that a sufficient training set is not always available for a specific user in a practical environment, our algorithm uses dynamic templates adaptation during the recognition process.

This paper is organized as follows: Section 2 proposes the GS-CPE framework along with its core services. Sections 3 and 4 describe the two key algorithms adopted in GS-CPE. Section 5 discusses related work and makes comparison between GS-CPE and other gesture interaction systems for smart homes. Section 6 concludes the paper.

2 GS-CPE framework

2.1 Personalization requirement analysis

Although many researches, including our previous work in [10, 11], claimed personalized gesture interactions, such previous work has not addressed the reasons why personalization should be considered as an important factor in designing gesture interaction systems for smart-home environments. Although researches in [12, 13] have revealed that users may have different preferences when performing gesture commands, their final objectives were to design user-independent gesture sets rather than to analyze the personalization requirements. Therefore, we start our work by answering this question following the same methodology as [12,13]. We first conducted an experiment to collect a gesture motion dataset consisting of 216 gesture instances for 8 camera-control commands from 27 participants (17 males and 10 females, aging from 18 to 22 years old) without prior experience of using gestural interfaces. Then, we analyzed the results using the normalized agreement rate $\mathrm{AR} \in [0, 1]$ proposed in our previous work [14]:

$$\mathrm{AR} = \frac{A_r - \frac{1}{|P_r|}}{1 - \frac{1}{|P_r|}}, \tag{1}$$

where $A_r \in [|P_r|^{-1}, 1]$ represents the absolute agreement rate calculated through the formula given in [15] for referent $r(1 \leqslant r \leqslant 8)$, and P_r represents the set containing all gesture motions for r. Because the range of AR is independent of the number of participants, AR is directly comparable across results from different datasets. Our experimental result shows an average AR of 0.21 (σ=0.05), which is smaller than that of 0.36 (σ=0.30) in [12] and that of 0.28 (σ=0.30) in [13]. As a smaller AR indicates a larger variance in gesture motion selection among different users, the result of our experiment suggests a stronger personalization requirement of gesture interactions in real-world smart-home environments.

2.2 Overall architecture

The architecture of GS-CPE is illustrated in Figure 1(a) and includes three major components:

1. The ontology database (OD) is the central database of GS-CPE, which stores the information of all objects in the system. For performance reasons, data from the OD is cached in the smart home controller and updated asynchronously during runtime.

2. The service portal (SP) is the core component where the smart home controller obtains services and applications. It uses the OD to collect information about objects and manage policies. It also provides functionalities to allow users to simulate and test services before downloading and executing them.

3. The smart home controller (SHC), a low-cost controller installed at home, is the major component at home-side. It acts as an intermediate agent between the cloud and the home-side devices, stores the service software as well as ontology data downloaded from the SP, and communicates with sensors and actuators through a lightweight event-driven architecture rather than a full-fledged service bus.

2.3 Core services and workflow

In the current version of GS-CPE, the gesture services, object management services and the event dispatch service are considered as "core services" as they implement the gesture recognition functionalities. Most APIs exposed by the core services are asynchronous methods. When invoked, each method generates an event with a unique transaction ID that is returned to the caller for further reference. In addition, the invocation generates a status code that indicates the execution status. The relationships among all core services and corresponding events are illustrated in Figure 1(b).

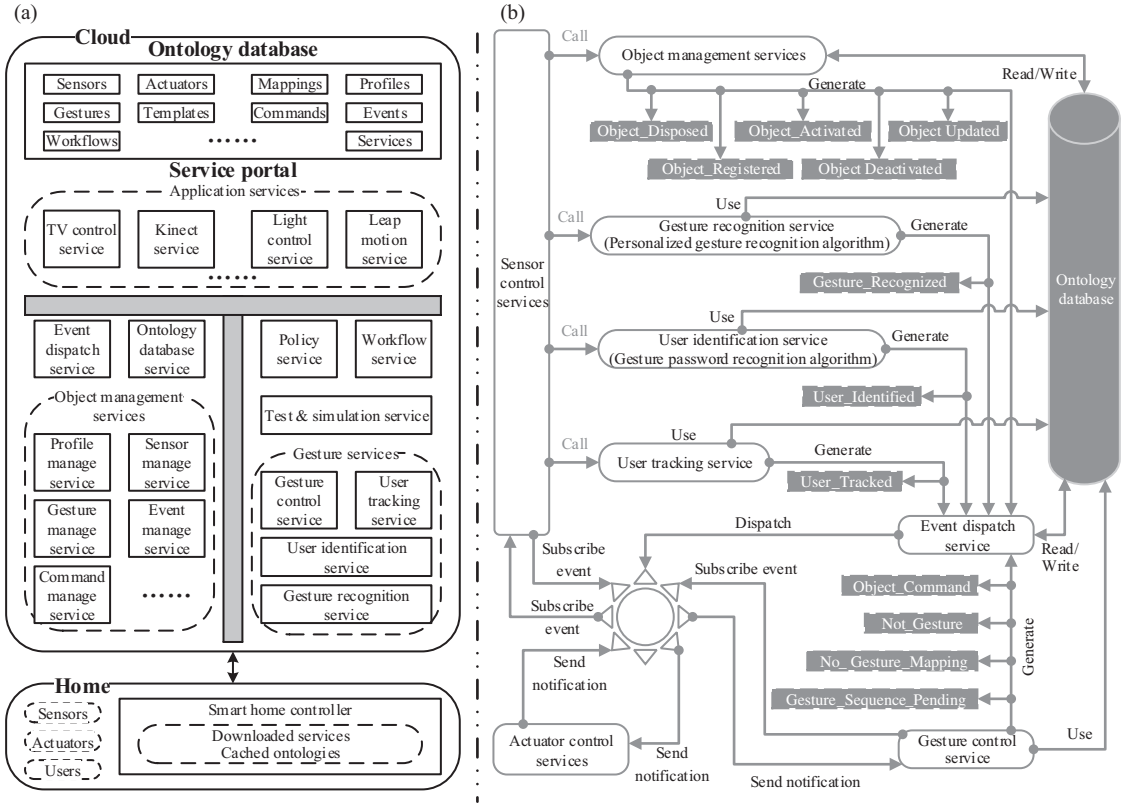

Figure 1 GS-CPE framework. (a) Architecture; (b) services and events.

Event dispatch service (EDS). In order to implement a lightweight event-driven architecture, GS-CPE adopts the EDS as the role of a publisher-subscriber of intermediate messages to disseminate events to their destinations correctly and efficiently. It exposes three synchronous methods: SubscribeEvent that allows a listener to subscribe to a specific event from an event generator, UnsubscribeEvent that allows a listener to unsubscribe from a previously subscribed event, and DispatchEvent that dispatches an activated event to all subscribers and write event logs into the OD.

Object management services (OMSs). The OMSs are collected services that maintain the registry of objects in GS-CPE, such as commands, gestures, devices, and user profiles, with each type of object having its own specific management service. The OMSs enable other services in GS-CPE to handle different types of objects using the same API set, which provides better dynamic extensibility and command management. Each object is identified by a unique object ID. All services in this group expose five asynchronous methods: Register that generates the Object_Registered event, Dispose that generates the Object_Disposed event, Activate that generates the Object_Activated event, Deactivate that generates the Object_Deactivated event, and Update that generates the Object_Updated event.

Sensor control services (SCSs). A SCS is a device-dependent service that converts raw data from a gesture acquisition device into filtered and quantified feature data. By designing a specific SCS for a specific type of gesture acquisition device, GS-CPE is able to support different gesture acquisition devices without modifying the core recognition algorithms.

Actuator control services (ACSs). An ACS is a device-dependent service that converts device-independent commands into actuator-specific commands. By designing different ACSs for different actuators, GS-CPE is able to support different actuators without modifying the other components.

Gesture recognition service (GRS). The GRS is responsible for recognizing personalized gestures produced by an identified user through the personalized gesture recognition algorithm discussed later in Section 4. The service exposes the Recognize method that generates the Gesture_Recognized event.

User identification service (UIS). To deliver smooth and personalized recognition experience

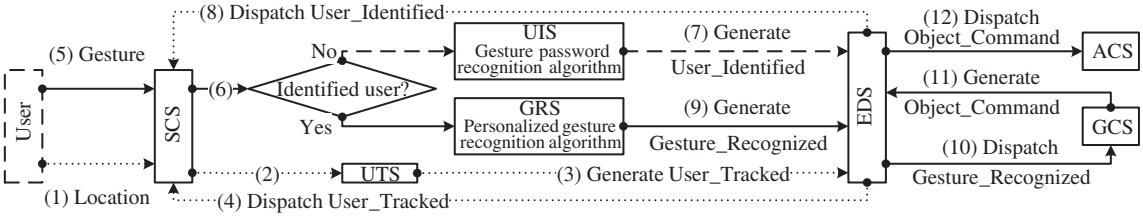

Figure 2 Typical workflow of GS-CPE.

for users, each user should be identified before they can perform gesture commands. An unidentified user needs to perform a series of single-handed writing gesture motions so that the UIS can classify the gesture motions into pre-defined symbol classes as a password through the gesture password recognition algorithm described in Section 3. Next, the UIS will search for a match in the OD to retrieve the user's gesture preference profile. When the authentication succeeds, the gesture preference profile's ID will be sent to the corresponding SCS through an event, so that the SCS can then call the GRS with the ID to make sure the user's consecutive gesture commands can be recognized through his own gesture preference profile for personalization. The service exposes the Identify method that generates the User_Identified event.

User tracking service (UTS). If a user's location can be continuously tracked, the framework can skip the identification process for the user after he roams to a new gesture acquisition device, which provides more seamless interaction experience. For example, given the limited sensing scope of a single Kinect sensor, a large room often needs multiple Kinect sensors to cover its entire space for continuous tracking. However, as each Kinect sensor performs tracking independently, a user may be assigned with different sensor-specific tracking IDs by multiple Kinect sensors. Therefore, it is the responsibility of the UTS to fuse the tracking data from multiple Kinect sensors so that the same user will be assigned with a unique global tracking ID. The tracking process is done by an improved version of the tracking algorithm described in our previous work [16], which will not be discussed in depth in this paper. UTS exposes the Track method that generates the User_Tracked event.

Gesture control service. The GCS handles classified gestures, converts them into device-independent commands, and generates corresponding events (Object_Command, Not_Gesture, Gesture_Sequence_Pending and No_Gesture_Mapping) to notify their subscribers (mostly ACSs).

Figure 2 illustrates a typical workflow of GS-CPE, where the Steps (1) to (4) (dotted lines) represent the optional user tracking workflow, Steps (5) to (8) (dashed lines) represent the user identification workflow, and Steps (5) and (6) along with Steps (9) to (12) represent the gesture recognition workflow. Note that the tracking process is optional for personalized gesture recognition to work properly, so that GS-CPE can also be deployed in places where tracking is not allowed because of privacy concerns or other reasons.

3 Gesture password recognition algorithm

In this section, we introduce the algorithm for gesture password recognition.

3.1 Feature preprocessing

In the algorithm, we choose the trajectory of a user's right hand as the gesture feature. Because the raw data from a Kinect sensor are 3-D floating-point position sequences, preprocessing can be applied in order to achieve higher recognition accuracy. First, as a gesture password is composed of symbols that are 2-D drawings articulated mostly in the x-y plane, the data of the z-axis can be safely discarded. Second, previous work [17, 18] found that the direction quantization approach offers both higher recognition accuracy and lower computational cost than simply using the raw data. Therefore, we preprocess the raw data in two steps: we first remove the z-axis data and then quantify the 2-D position sequence into an integer direction sequence using the direction quantization scheme proposed in our previous work [11]

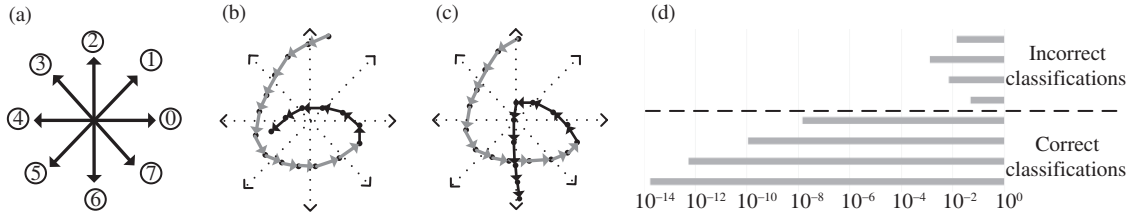

Figure 3 The direction quantization scheme, examples of trajectory sequences and likelihood comparison. (a) Quantization scheme; (b) trajectory of "6"; (c) trajectory of "4"; (d) comparison of second maximum likelihood ratio.

which is shown in Figure 3(a). After the preprocessing phase, the trajectory sequence is represented as an integer direction sequence $\{\theta_0, \theta_1, \ldots, \theta_{n-1}\}$.

3.2 Algorithm design

The purpose of gesture password recognition is to identify users by recognizing user-independent symbol writing gestures from uncertain users, for which statistical-model classifiers are widely used. Among these models, the Hidden Markov Model (HMM) [19] has been found to be an effective way to classify such gestures. Therefore, we first tried to employ HMM as the main classifier of the recognition algorithm. However, when evaluating the HMM classifier for our real-world digit symbol gesture dataset, we found that its recognition accuracy was not as high as we were expecting. Our finding reveals that in a practical environment the HMM classifier does not deal well with variations in the generation of symbol gesture trajectories when the training dataset is not large enough. For example, Figure 3(b) and (c) show the trajectories of two digit symbols. The start of both feature sequences (i.e., gray arrows) is the same, and the difference occurs only at the trailing parts (i.e., black arrows). Although the two symbols are visually distinguishable, the system may mistakenly regard them as the same symbol because the likelihood values between the two symbols are extremely closer than others. Denote $\boldsymbol{l}^g = (l_0^g, l_1^g, \ldots, l_9^g)$ is the vector containing likelihood values between input gesture g and HMM models "0" to "9", we compute the likelihood ratio vector $\mathbf{lr}^g = (\frac{l_0^g}{\max(\boldsymbol{l}^g)}, \frac{l_1^g}{\max(\boldsymbol{l}^g)}, \ldots, \frac{l_9^g}{\max(\boldsymbol{l}^g)})$ for g and show values of the secondary maximum likelihood ratio $\mathrm{secmax}(\mathbf{lr}^g)$ from some test cases in Figure 3(d). From Figure 3(d) one can see that, the values of $\mathrm{secmax}(\mathbf{lr}^g)$ for incorrect classifications are much larger than those for correct classifications, and we further found that more than half of the incorrect classifications have such significantly large $\mathrm{secmax}(\mathbf{lr}^g)$. Therefore, it is reasonable to make the following assumptions:

1. For a given gesture input g, if $\exists h \in \mathbb{H}$ so that $\frac{l_h^g}{\max(\boldsymbol{l}^g)} \geqslant \epsilon$, then we have $\frac{P(H_g \neq L_g|\epsilon)}{P(H_g = L_g|\epsilon)} > 1$;
2. For a given gesture input g, if $\nexists h \in \mathbb{H}$ so that $\frac{l_h^g}{\max(\boldsymbol{l}^g)} \geqslant \epsilon$, then we have $\frac{P(H_g \neq L_g|\epsilon)}{P(H_g = L_g|\epsilon)} \leqslant 1$,

where H_g denotes the classification result from HMM classifier, L_g denotes the real label, $\mathbb{H}$ denotes the set of all HMM models, h denotes a HMM model, ϵ is a constant, and $P(H_g \neq L_g|\epsilon)$ and $P(H_g = L_g|\epsilon)$ denote the probability of incorrect and correct classifications with respect to a given ϵ respectively.

Algorithm 1 The HMM-GSS algorithm

Input: Input trajectory g, threshold value ϵ, GSS standard template set $\mathbb{ST}$;
Output: Label;

```
H_g, lr^g ← Classification result and likelihood ratio vector of input gesture g from the HMM classifier;
if secmax(lr^g) < ε then
    Label ⇐ H_g;
else
    R = {h | h ∈ H ∧ l_h^g / max(l^g) ⩾ ε};
    GT = {t_i | t_i ∈ ST ∧ i ∈ R};
    Label ⇐ S_g, which is the classification result of input gesture g from the GSS classifier according to ST;
end if
```

Based on the assumptions, we propose a two-phase cascading classifier described in Algorithm 1. During the first phase, when $\mathrm{secmax}(\mathbf{lr}^g) < \epsilon$ satisfies, the classifier accepts H_g as the final result for a gesture input. Otherwise, the input should be sent to the second-level template-matching classifier for further

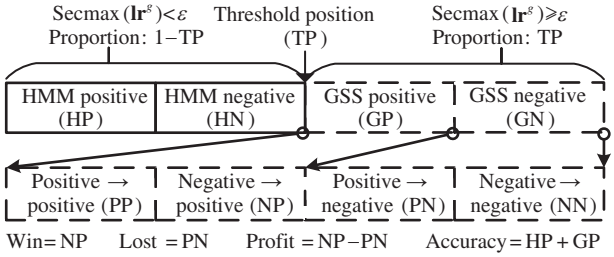

Figure 4 The relationship between indicators and TP.

classification, which may be helpful to mitigate some misclassifications introduced by HMM. Commonly used template-matching classifiers include golden section search (GSS) [20], Hungarian [21] and dynamic time warping (DTW) [22]. As GSS has good performance in recognizing unistroke gestures [23], which is suitable to handle the shape information of a symbol writing gesture trajectory, we consider GSS as the second-level classifier and denote the algorithm as HMM-GSS.

As GSS is a template-matching algorithm, template selection has direct impact on its recognition accuracy. Thus the template set should be carefully selected in order to maximize the recognition accuracy of the GSS classifier. Given a pre-collected training dataset $\mathbb{T} = \{T_i^k | k \in [1, n], i \in [1, n_k]\}$ (where n is the number of different symbol gestures, T_i^k is the ith data sample of symbol gesture k, and $n_k > 0$ is the amount of data samples for symbol gesture k), the standard template set $\mathbb{ST}$ is defined by (2):

$$\mathbb{ST} = \left\{ \mathrm{ST}_k = \arg\min_{j \in [1, n_k]} \frac{1}{n_k} \sum_{i=1}^{n_k} \mathrm{Distance}(T_j^k, T_i^k) | k \in [1, n], T_i^k \in \mathbb{T}, T_j^k \in \mathbb{T} \right\}, \tag{2}$$

where ST_k is the standard template of symbol gesture k and $\mathrm{Distance}(T_j^k, T_i^k)$ calculates the distance between T_j^k and T_i^k according to some distance metric. Moreover, ST_k can be updated adaptively when new data is available.

3.3 Parameter selection

According to Algorithm 1, the value of ϵ directly affects the number of inputs that should be classified by the GSS classifier (GSS-set) or is only classified by the HMM classifier (HMM-set), and eventually impacts on the final recognition accuracy. To find the best value for ϵ, we first define four indicators that are illustrated in Figure 4: HP and HN that represent the percentages of correctly and incorrectly classified gestures in the HMM-set respectively, and GP and GN that represent the corresponding percentages in the GSS-set. Obviously, the four indicators satisfy that HP+HN+GP+GN=1 and the overall recognition accuracy of the algorithm is HP+GP. Therefore, in order to maximize the overall recognition accuracy, one should solve the following optimization problem:

$$\arg\max_{\epsilon \in [0,1]} (\mathrm{HP} + \mathrm{GP}). \tag{3}$$

However, because the functional relationship between HP+GP and ϵ is unknown, the problem cannot be directly solved, which means that ϵ should be adjusted according to the training dataset $\mathbb{G}$ with an iterative method through the following steps:

1. $\forall g \in \mathbb{G}$, calculate $\mathbf{lr}^g$ and H_g, and construct $\mathbb{LR} = \{\mathrm{lr} | \mathrm{lr} \in \mathbf{lr}^g, g \in \mathbb{G}\}$;
2. $\forall \mathrm{lr} \in \mathbb{LR}$, if lr satisfies $\frac{P(H_g \neq L_g | \mathrm{lr})}{P(H_g = L_g | \mathrm{lr})} > 1$, then add lr into the candidate set $\boldsymbol{\epsilon}$;
3. $\forall \epsilon' \in \boldsymbol{\epsilon}$, get all S_g that satisfies $\mathrm{secmax}(\mathbf{lr}^g) \geqslant \epsilon'$ and calculate the corresponding HP and GP;
4. Choose the $\epsilon' \in \boldsymbol{\epsilon}$ that maximizes the value HP+GP as the best estimation for ϵ.

As the absolute value of the best estimation for ϵ depends on the given dataset and lacks of physical significance and intuitiveness, it is better to choose a more intuitive parameter that has a functional relationship with ϵ. One can notice that, each ϵ partitions a dataset into two groups: one group is sent to the GSS classifier while the other group is not. Therefore one can define a threshold position (TP)

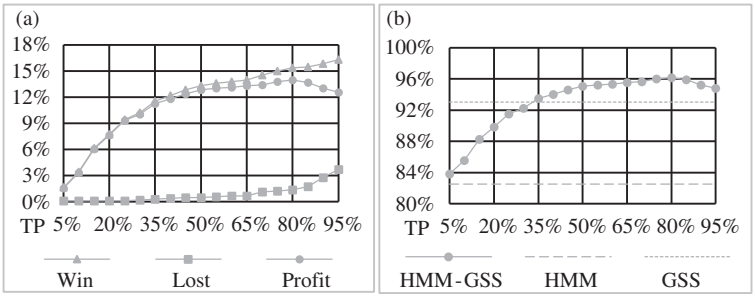

Figure 5 The experiment results of HMM-GSS algorithm. (a) Average win/lost/profit; (b) average accuracy comparison.

that represents the percentage of data to be classified by the GSS classifier. Obviously, the functional relationship between ϵ and TP exists, and TP is intuitive. We will use TP in the evaluation part.

3.4 Evaluation and analysis

The dataset we used contains 10 digit symbol writing gestures from 14 participants using Kinect sensors. Each participant performs each writing gesture at least four times and the final dataset contains 525 valid samples after removing some invalid samples. Then the dataset was randomly split into five cross-validation groups, with each group containing a training set of 60% samples and a test set of 40% samples.

For accurate evaluation of the algorithm, we define three indicators of win, lost and profit to reflect the performance change between the HMM-only single-level classifier and the two-level classifier. To calculate the three indicators, another four intermediate indicators of PP, NP, PN and NN are introduced and the relationships of all indicators are illustrated in Figure 4. Figure 5 illustrates the relationships between TP and win/lost/profit (Figure 5(a)) as well as the overall trend of accuracy (Figure 5(b)). The figure illustrates that no matter how TP is set, the HMM-GSS classifier outperforms the HMM-only classifier in terms of the benefit of a positive profit. Although both win and lost rise as TP increases, the win (1.67%→13.95%) rises much faster than the lost (0.09%→0.65%) when TP is under 70%. Then the rising rate of lost increases so that the profit shows as a unimodal function with the peak value (13.95%) appearing when TP is set to 80%. At this point, the HMM-GSS classifier achieves a maximum accuracy of 96.19%, which is not only higher than the HMM-only classifier (82.24%) but also higher than the GSS-only classifier (93.02%). As the accuracy of GSS-only classifier is higher than the HMM-only classifier in our dataset, the set of TP causes the algorithm to accept more GSS-classified results than HMM-classified results. We also evaluate the time performance of the HMM-GSS classifier, and the results show that the classifier produces the classification result for one gesture input in less than 4 ms in our dataset.

4 Personalized gesture recognition algorithm

In this section, we describe the personalized gesture recognition algorithm in detail.

4.1 Feature selection and preprocessing

The algorithm accepts two types of features for gesture recognition: the acceleration data and the position data. The acceleration data are provided by sensors equipped with accelerometers such as a smartphone or the Nintendo Wii Remote controller. Each data sample provided by an accelerometer is a 3-axis vector (a_x, a_y, a_z) that contains acceleration values for each axis. Therefore, if a gesture is produced with only one hand, the feature vector can be defined as $\boldsymbol{f}_a^s = (a_x^{\mathrm{l}}, a_y^{\mathrm{l}}, a_z^{\mathrm{l}})$ or $\boldsymbol{f}_a^s = (a_x^{\mathrm{r}}, a_y^{\mathrm{r}}, a_z^{\mathrm{r}})$ for each data sample. If a gesture is bimanual, the feature vector can be defined as $\boldsymbol{f}_a^d = (a_x^{\mathrm{l}}, a_y^{\mathrm{l}}, a_z^{\mathrm{l}}, a_x^{\mathrm{r}}, a_y^{\mathrm{r}}, a_z^{\mathrm{r}})$, where $(a_x^{\mathrm{l}}, a_y^{\mathrm{l}}, a_z^{\mathrm{l}})$ and $(a_x^{\mathrm{r}}, a_y^{\mathrm{r}}, a_z^{\mathrm{r}})$ represent the acceleration data of the left hand and the right hand respectively.

The position data are provided by sensors equipped with video cameras, such as a Microsoft Kinect sensor or a Leap Motion sensor. In this paper, we focus on the Kinect sensor. As each data sample from the Kinect sensor contains 20 three-dimensional vectors, only a subset of the returned vectors with the

most significant movements are chosen as the gesture feature to reduce the complexity of representation and recognition process. Then the relative position between a hand position and the spine position is used to remove the influence of user's absolute standing position. The feature vector of each data sample is defined as $\boldsymbol{f}_p^s = (v_x^{\mathrm{l}} - v_x^{\mathrm{s}}, v_y^{\mathrm{l}} - v_y^{\mathrm{s}}, v_z^{\mathrm{l}} - v_z^{\mathrm{s}})$ or $\boldsymbol{f}_p^s = (v_x^{\mathrm{r}} - v_x^{\mathrm{s}}, v_y^{\mathrm{r}} - v_y^{\mathrm{s}}, v_z^{\mathrm{r}} - v_z^{\mathrm{s}})$ for a single-handed gesture and $\boldsymbol{f}_p^d = (v_x^{\mathrm{l}} - v_x^{\mathrm{s}}, v_y^{\mathrm{l}} - v_y^{\mathrm{s}}, v_z^{\mathrm{l}} - v_z^{\mathrm{s}}, v_x^{\mathrm{r}} - v_x^{\mathrm{s}}, v_y^{\mathrm{r}} - v_y^{\mathrm{s}}, v_z^{\mathrm{r}} - v_z^{\mathrm{s}})$ for a bimanual gesture, where $(v_x^{\mathrm{l}}, v_y^{\mathrm{l}}, v_z^{\mathrm{l}})$, $(v_x^{\mathrm{r}}, v_y^{\mathrm{r}}, v_z^{\mathrm{r}})$ and $(v_x^{\mathrm{s}}, v_y^{\mathrm{s}}, v_z^{\mathrm{s}})$ represent the corresponding positions of the left hand, right hand and the spine.

As the raw data may contain noise that can degrade the recognition accuracy, each sample is processed with a moving average filter. Also, a quantization process is applied to the filtered data to reduce time-consuming floating point calculation, which quantizes the floating-point values into integers within $[-31, +31]$ according to the technique proposed in our previous work [10].

4.2 Algorithm design

According to [24], a DTW classifier can achieve better recognition accuracy than a HMM classifier with fewer training samples. As the DTW classifier employs the template-matching mechanism rather than a statistics model, using a very small training dataset (even only one sample) for each gesture type may lead to relatively high recognition accuracy if the templates are appropriately selected. Therefore, DTW has been widely used for gesture recognition [25, 26]. However, conventional DTW has the following limitations that prevent it from achieving higher recognition accuracy in many situations:

1. Conventional DTW adopts only fixed templates for each gesture type, which limits its performance for gesture instances with large variations.

2. Implementing rejection is challenging for conventional DTW because only likelihood values can be used to determine whether a recognition result should be rejected or not. However, choosing an appropriate rejection likelihood threshold is difficult because it may vary significantly for different datasets.

Therefore, to overcome these limitations, we proposed an enhanced DTW to implement the personalized gesture recognition algorithm. The enhancements are summarized as follows:

1. We introduce a standard template selection process during the bootstrap phase, and a dynamic template adaptation process for updating standard templates during the recognition phase;

2. We introduce a new rejection determination criterion using multiple input sources instead of the rejection likelihood threshold.

When using multiple templates for each gesture type, there are several ways to select the best-matching gesture. In this paper, we consider three template matching criteria: nearest neighbor (NN), K-nearest neighbor (KNN) and nearest group (NG).

4.2.1 *Standard template selection and adaptation*

Before a DTW classifier can be used for classification, it is important to select the standard templates for each gesture type to maximize the recognition accuracy. Suppose that each gesture type has l ($l \geqslant 1$) templates and the initial sample set $\boldsymbol{T} = \{T_1, T_2, \ldots, T_m\}$ of this gesture type contains m ($m \geqslant l$) samples, we can construct a set $\boldsymbol{d}$ in which each element is the sum of DTW distances from one sample to all the other samples calculated according to (4):

$$\boldsymbol{d} = \left\{ d_j = \sum_{i=1}^{m} \mathrm{DTW}(T_i, T_j) | j \in [1, m] \right\}. \tag{4}$$

Then the l gesture samples in $\boldsymbol{T}$ corresponding to the smallest l values from set $\boldsymbol{d}$ are the initial standard templates of each gesture type. During the recognition process, when the number of rejected inputs for a gesture type reaches a specific amount, the dynamic template adaptation process is invoked to update the standard templates for this gesture type to reflect the user's most recent gesture preference.

4.2.2 *Rejection determination*

Previous research [25] has showed that rejecting unreliable recognition results can improve the recognition accuracy. However, it is not straightforward to determine whether a recognition result should be rejected or not. Common criteria such as likelihood or distance values may vary significantly for different datasets even when they are normalized. However, if a gesture motion is captured by multiple sensors simultaneously, we can get one recognition result for each input source independently. If results from all input sources are the same, the result can be accepted. Otherwise, results should be rejected or further examined using other criteria. In this paper, for simplification purposes, the proposed algorithm only considers two different input sources (position and acceleration).

4.2.3 *Algorithm description*

The details of the algorithm are described in Algorithm 2. Suppose there are n gesture types with each type having l standard templates, then the algorithm takes the user-specific standard template set $\mathbb{T}^u = \{\boldsymbol{T}_i^u = \{\langle \mathbf{ta}_k^i, \mathbf{tp}_k^i \rangle | k \in [1,l]\} | i \in [1,n]\}$ (where u is the gesture profile ID of a specific user) and the input gesture sample set $\mathbb{G} = \{\mathbf{Ga}, \mathbf{Gp}\}$ as input (postfix "**a**" means "acceleration" and postfix "**p**" means "position"), and outputs the recognized gesture label ($1 \sim n$) or 0 in case of rejection.

Algorithm 2 The multiple-source DTW (MS-DTW) algorithm

Input: $\mathbb{T}^u, \mathbb{G}$;
Output: L;
Initialize $\mathbf{Da} = \{\mathrm{Da}_k^i = \mathrm{DTW}(\mathbf{Ga}, \mathbf{ta}_k^i)\}, \mathbf{Dp} = \{\mathrm{Dp}_k^i = \mathrm{DTW}(\mathbf{Gp}, \mathbf{tp}_k^i)\}, \mathbf{Tr}_i \leftarrow \phi$, where $i \in [1,n], k \in [1,l]$;
//The template matching process;
if using the K-nearest neighbor criterion or nearest neighbor criterion **then**
  $\mathbf{da} \leftarrow \{\mathrm{da}_i | i \in [1,K]\}$ where da_i belongs to the minimal K values in **Da**;
  $\mathbf{dp} \leftarrow \{\mathrm{dp}_i | i \in [1,K]\}$ where dp_i belongs to the minimal K values in **Dp**;
  $\mathrm{Label}_a \leftarrow$ The majority class in **da**;
  $\mathrm{Label}_p \leftarrow$ The majority class in **dp**;
else
  $\mathbf{da} \leftarrow \{\mathrm{da}_i = \frac{1}{l}\sum_{k=1}^{l} \mathrm{Da}_k^i | i \in [1,n] \wedge \mathrm{Da}_k^i \in \mathbf{Da}\}$;
  $\mathbf{dp} \leftarrow \{\mathrm{dp}_i = \frac{1}{l}\sum_{k=1}^{l} \mathrm{Dp}_k^i | i \in [1,n] \wedge \mathrm{Dp}_k^i \in \mathbf{Dp}\}$;
  $\mathrm{Label}_a \leftarrow \mathrm{argmin}_{i \in [1,n]}\{\mathrm{da}_1, \mathrm{da}_2, \ldots, \mathrm{da}_n\}$;
  $\mathrm{Label}_p \leftarrow \mathrm{argmin}_{i \in [1,n]}\{\mathrm{dp}_1, \mathrm{dp}_2, \ldots, \mathrm{dp}_n\}$;
end if
//The rejection determination process;
if $\mathrm{Label}_a = \mathrm{Label}_p$ **then**
  $L \leftarrow \mathrm{Label}_a$;
else
  $L \leftarrow 0$;
end if
//The dynamic template adaptation process;
if $L \neq 0$ **then**
  $\mathbf{Tr}_L \leftarrow \mathbf{Tr}_L \cup \{\mathbb{G}\}, \mathrm{rc}_L \leftarrow 0$;
else
  if received correct label g **and** $|\mathbf{Tr}_g| > 0$ **then**
    Update $\boldsymbol{T}_g^u$ using $\mathbf{Tr}_g \cup \boldsymbol{T}_g^u$ through the same way of selecting initial standard templates;
    $\mathbf{Tr}_g \leftarrow \phi$;
  end if
end if

If the average length of standard templates and input data are p and q respectively, then the time complexity for calculating all DTW distances is $O(pqnl)$. After calculation, the template matching process normally takes $O(nl)$, $O(nl \log K)$ and $O(nl)$ for NN, KNN and NG criteria respectively (for large datasets, hash-based methods [27, 28] may be used to speed up the NN or KNN process). The dynamic template adaptation process takes $O(pql + l \log K)$ to complete, while the rejection determination process takes a constant time to complete. As l is normally a small integer no more than 5, while p and q have the same magnitude, the time complexity for this algorithm can be simplified as $O(np^2 + n)$, which indicates

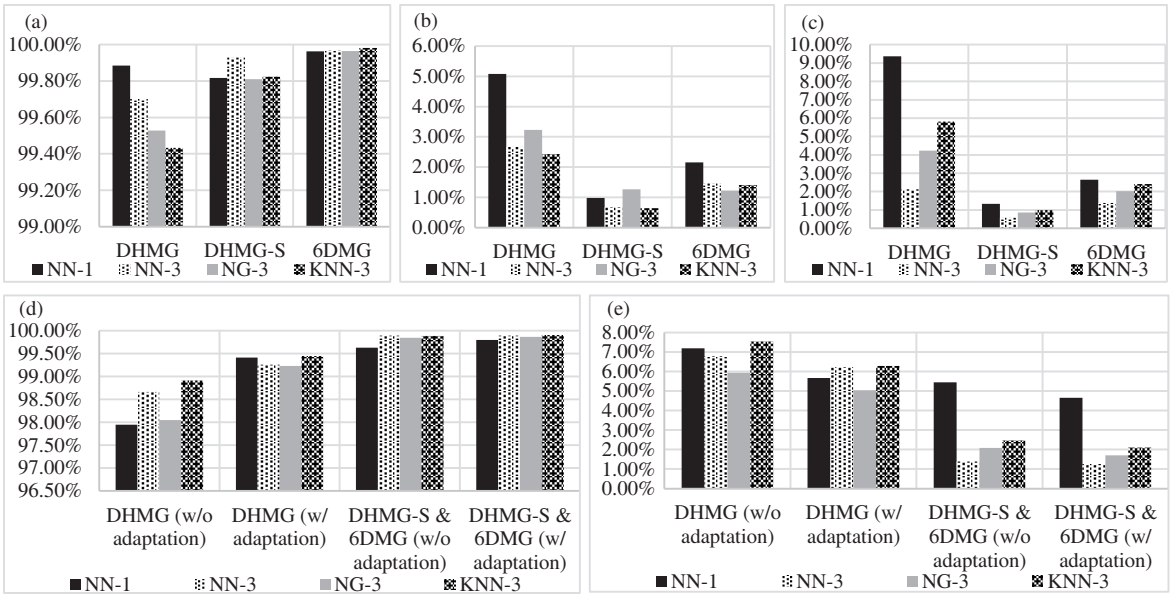

Figure 6 Theoretical and practical performance of MS-DTW. (a) Theoretical accuracy comparison; (b) accuracy increment by multiple-source; (c) theoretical rejection comparison; (d) practical accuracy comparison w/ and w/o template adaptation; (e) practical rejection comparison w/ and w/o template adaptation.

that the most time consumption of this algorithm is still the DTW distance calculation process and the improvement has little impaction on the overall time complexity.

4.3 Evaluation and analysis

4.3.1 *Dataset*

We evaluate our algorithm on two different gesture sets: the 6DMG (6D Motion Gesture) [29] dataset published by Georgia Institute of Technology and the DHMG (Dual-Handed Motion Gesture) dataset we collected in [10]. The 6DMG dataset contains 20 different single-handed gestures from 28 participants with each participant performing one gesture for 10 times in a single day, resulting a total number of 5600 gesture samples. Each sample in the 6DMG dataset contains a 3-D position vector from the WorldViz PPT-X4 sensor and a 3-axis acceleration vector from the Wii Remote sensor. The DHMG dataset contains 8 different bimanual gestures from 6 participants with each participant performing one gesture for 50 times in 5 different days (10 times per day), resulting in a total number of 2400 gesture samples. Each sample in the DHMG dataset contains a feature vector $\boldsymbol{f}_p^d$ from the Kinect sensor and a feature vector $\boldsymbol{f}_a^d$ from the Wii Remote sensor. As the DHMG dataset consists of data from 5 days, in order to evaluate the performance difference between data from different days and data from the same day, we created a new dataset called DHMG-S that contains 5 sub-datasets. The ith sub-dataset in DHMG-S contains only the data from the ith day in the original DHMG dataset.

4.3.2 *Theoretical performance*

We first evaluate the theoretical performance of the MS-DTW algorithm without dynamic template adaptation using different template matching criteria in multiple metrics, including theoretical accuracy and rejection for different datasets, and the error decrement by using multiple sources over a single source. In this evaluation, we ran the experiments for several times and at each time the initial standard templates were randomly chosen from the dataset. The average results are shown in Figure 6(a)–(c), where the accuracy is calculated excluding the rejected data. Although all criteria achieve accuracy rates higher than 99% with tiny variation, the difference of rejection rates among different criteria varies more significantly. All the multiple-template matching criteria demonstrate lower rejection rate than the single-template criterion (NN-1), and among which the nearest neighbor criterion with 3 standard

templates (NN-3) reaches the lowest rejection rate compared to other two criteria (NG-3, KNN-3). Such a difference in the rejection metric becomes more obvious when the input gestural data presents higher variance: the rejection rate with the DHMG dataset shows more variance than those with the other two datasets. This fact occurs because the variation of the gestural samples in the DHMG dataset is much larger than that of the other two datasets. Even in the worst case, all multiple-template matching criteria yield better performance than NN-1 in term of the rejection rate (<6.0% vs 9.3%). For all tested datasets, the multiple-template matching criteria achieved an accuracy rates of 99.8% with a rejection rate of 3.0% in average.

In comparison with the performance of the conventional DTW using a single input source, the MS-DTW successfully increases the accuracy rate significantly for both the DHMG dataset and the 6DMG dataset. Despite of the relative narrow gain in the case of the DHMG-S dataset, multiple sources tend to have a positive effect on the recognition accuracy of MS-DTW because unreliable recognition results are discarded. In average, MS-DTW with multiple-template matching criteria achieves an accuracy improvement of 1.7% for all tested datasets.

4.3.3 *Practical scenario performance*

The theoretical performance of the MS-DTW algorithm may not be achieved in real scenarios, because in many cases it is impossible to pre-collect a gesture dataset for each user. Therefore, we tested the performance of MS-DTW in real circumstances by setting the initial standard templates of a gesture type to each user's corresponding first l data samples. We then evaluated our algorithm's performance with and without dynamic template adaptation. The experimental results are shown in Figure 6(d) and (e). The figure illustrates that when used in a practical scenario, the algorithm's performance reduction depends on the data variation of the dataset. For the DHMG-S and 6DMG datasets (small variation), the rate reduction in the average accuracy is very small ($\leqslant$0.1%) either with or without template adaptation, while the increase in the average rejection rate is 1.32% without template adaptation and 0.9% with template adaptation. However, for the DHMG dataset (large variation), the MS-DTW's performance degrades more than the previous case: the rate reduction in the average accuracy becomes 1.25% and the increase in the rejection rate becomes 1.47%. To remedy the negative impact caused by variation, the dynamic template adaptation is implemented in the MS-DTW. Using adaptation, the reduction rate in the accuracy drops from 1.25% to 0.3% and the increase in the rejection rate drops from 1.47% to 0.41%. Even without the adaptation, multiple-template matching criteria can still enable the MS-DTW to achieve an average accuracy rate of 98.5% for the DHMG dataset. During the above experiments, we also evaluated the algorithm's time performance and found that even in the worst case, the MS-DTW algorithm ran within 4 ms (the DHMG dataset) or 10 ms (the 6DMG dataset) for one gesture input through a PC with a Quad-core 3.50 GHz CPU.

5 Related work and comparison

5.1 Related work

Gesture recognition has been widely investigated for human-computer interaction [30]. Various types of sensors are used to capture gestures, such as the smartphone [31], the Nintendo Wii Remote controller [25, 32], the Microsoft Kinect sensor [2, 33], the Leap Motion sensor [7] and even the WiFi Access Point [34]. Gesture recognition devices [3], libraries [32], frameworks [35] and applications [2, 5, 7, 33] are created to provide interactions in either cyber spaces or physical spaces, among which some research efforts focus on gesture interactions in smart-home environments. Panger [2] studied the usage of Kinect in controlling household appliances in kitchen. Our previous research in [7] utilized Leap Motion to capture gestures for smart TV control purpose. Pan et al. [3] modified the Wii Remote controller for controlling smart-home appliances through gestures, while Kühnel et al. [5] and van Seghbroeck et al. [35] utilized the smartphone and the Sun SPOT wireless sensor for similar tasks respectively. And Pu et al. [34]

Table 1 Comparison of systems designed for smart-home environments

Criterion	GS-CPE	GeeAir [3]	GPWS [9]	I'm home [5]	WS-Gesture [35]	WiSee [34]
Driven style	Event-driven	Data-driven	Event-driven	N/A	Message-driven	N/A
User identification	Yes	No	No	No	No	No
Personalization	Yes	No	No	No	No	No
Multiple input sources	Yes	No	No	No	No	No
Acquisition device	Kinect Wii remote	Modified Wii remote	Wii remote	Smartphone	Sun SPOT	WiFi AP
Recognition algorithm	MS-DTW	FDSVM	DTW	FastDTW	HMM + k-means	Simple match
Overall accuracy	98.5%	96.4%	96%	90.8%	90%	94%
Response time	$\leqslant$30 ms	$\leqslant$60 ms	$\leqslant$500 ms	N/A	$\leqslant$200 ms	N/A

discovered the possibility of using WiFi signals for gesture recognition at home.

Although user identification is important in a practical environment [36], most previous research efforts put emphasis on gesture recognition only and did not implement user identification. Some work have partially addressed this problem by introducing gesture-based user authentication and identification [25, 37]. However, these systems either focused only on identification or supported only fixed gesture models rather than user-defined personalized gestures, and thus they could not provide a full personalized gesture interaction solution.

Recently, SOA and ontologies have been employed for gesture recognition [9, 35, 38] to provide more flexibility and extensibility. Among this work, WS-Gesture [35] and GPWS [9] are two representatives of different approaches: WS-Gesture adopts the enterprise-level service-oriented architecture, while GPWS uses a lightweight event-driven framework. As devices in a smart home often have limited computing resources to afford the heavy computation required by enterprise-level services, the event-driven approach used by GPWS is more appropriate for developing a gesture interaction framework for smart homes.

5.2 Comparison

We choose a series of key evaluation criteria inspired from [39] to compare GS-CPE with other gesture interaction systems designed for smart homes in Table 1. The highlights of GS-CPE include:

- GS-CPE is the only system that provides full personalization support with user identification.
- GS-CPE is the only system that supports two types of acquisition devices (and is capable for more) to provide multiple input sources for gesture recognition.
- GS-CPE reduces response time by applying personalized profiles (that filters templates during matching) and by implementing more efficient algorithms (that reduces the computation overhead).
- GS-CPE achieves higher recognition accuracy than other systems through personalization and dynamic template adaptation.

6 Conclusion

This paper proposed GS-CPE, a new service-oriented framework for personalized gesture interaction in cyber-physical smart-home environments. Our framework adopts two new algorithms: the HMM-GSS algorithm for user identification and the MS-DTW algorithm for personalized gesture recognition. The proposed HMM-GSS algorithm, which recognizes gesture passwords for user identification, delivers an accuracy rate of 96.2%, which is higher than the accuracy rates of both conventional HMM and GSS algorithms. And the proposed MS-DTW algorithm achieves an average improvement of accuracy rates by 1.7% using two input sources, and the experimental results also show that its average accuracy rate in practical scenarios is 98.5% with template adaptation enabled. The comparison between GS-CPE and other gesture interaction systems reveals that GS-CPE has several advantages including supporting multiple input sources and full personalization, higher recognition accuracy and smaller response time.

Acknowledgements This work was supported by National High Technology Research and Development Program of China (Grant No. 2013AA01A210), State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2013ZX-03), and National Natural Science Foundation of China (Grant No. 61532004). Vatavu also acknowledges support from the project "Integrated Center for Research, Development and Innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for Fabrication and Control" (Grant No. 671/09.04.2015), Sectorial Operational Program for Increase of the Economic Competitiveness, co-funded from the European Regional Development Fund.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Bernhaupt R, Obrist M, Weiss A, et al. Trends in the living room and beyond: results from ethnographic studies using creative and playful probing. ACM CIE, 2008, 6: 5
2 Panger G. Kinect in the kitchen: testing depth camera interactions in practical home environments. In: Proceedings of the CHI Extended Abstracts on Human Factors in Computing Systems. New York: ACM, 2012. 1985–1990
3 Pan G, Wu J H, Zhang D Q, et al. GeeAir: a universal multimodal remote control device for home appliances. Pers Ubiquitous Comput, 2010, 14: 723–735
4 Vatavu R D. Point & click mediated interactions for large home entertainment displays. Multimed Tools Appl, 2012, 59: 113–128
5 Kühnel C, Westermann T, Hemmert F, et al. I'm home: defining and evaluating a gesture set for smart-home control. Int J Hum-Comput Stud, 2011, 69: 693–704
6 Vatavu R D. A comparative study of user-defined handheld vs. freehand gestures for home entertainment environments. J Ambient Intell Smart Environ, 2013, 5: 187–211
7 Vatavu R D, Zaiti I A. Leap gestures for TV: insights from an elicitation study. In: Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video. New York: ACM, 2014. 131–138
8 Li W, Lee Y H, Tsai W T, et al. Service-oriented smart home applications: composition, code generation, deployment, and execution. Serv Oriented Comput Appl, 2012, 6: 65–79
9 Vatavu R D, Chera C M, Tsai W T. Gesture profile for web services: an event-driven architecture to support gestural interfaces for smart environments. In: Ambient Intelligence. Berlin: Springer-Verlag, 2012. 161–176
10 Lou Y H, Wu W J. A real-time personalized gesture interaction system using Wii remote and Kinect for tiled-display environment. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering. Skokie: KSI, 2013. 131–136
11 Zhang H K, Wu W J, Lou Y H. A personalized gesture interaction system with user identification using Kinect. In: PRICAI 2014: Trends in Artificial Intelligence. Berlin: Springer, 2014. 614–626
12 Vatavu R D. User-defined gestures for free-hand TV control. In: Proceedings of the 10th European Conference on Interactive TV and Video. New York: ACM, 2012. 45–48
13 Wobbrock J O, Morris M R, Wilson A D, et al. User-defined gestures for surface computing. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM, 2009. 1083–1092
14 Vatavu R D, Wobbrock J O. Formalizing agreement analysis for elicitation studies: new measures, significance test, and toolkit. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM, 2015. 1325–1334
15 Wobbrock J O, Aung H H, Brandon R, et al. Maximizing the guessability of symbolic input. In: Proceedings of the CHI Extended Abstracts on Human Factors in Computing Systems. New York: ACM, 2005. 1869–1872
16 Lou Y H, Yao T, Chen Y Q, et al. A novel scheme of ROI detection and transcoding for mobile devices in high-definition videoconferencing. In: Proceedings of the 5th Workshop on Mobile Video. New York: ACM, 2013. 31–36
17 Wang Y W, Yang C, Wu X, et al. Kinect based dynamic hand gesture recognition algorithm research. In: Proceedings of the 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, Nanchang, 2012. 274–279
18 Zhu H M, Pun C M. Real-time hand gesture recognition form depth image sequences. In: Proceedings of the 9th International Conference on Computer Graphics, Imaging and Visualization, Hsinchu, 2012. 49–52
19 Moni M A, Shawkat Ali A B M. HMM based hand gesture recognition: a review on techniques and approaches. In: Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, 2009. 433–437
20 Kiefer J. Sequential minimax search for a maximum. Proc American Math Soc, 1953, 4: 502–506
21 Vatavu R D, Anthony L, Wobbrock J O. Gestures as point clouds: a $P recognizer for user interface prototypes. In: Proceedings of the 14th ACM International Conference on Multimodal Interaction. New York: ACM, 2012. 273–280
22 Myers C S, Rabiner L R. A comparative study of several dynamic time-warping algorithms for connected word recognition. Bell Syst Tech J, 1981, 60: 1389–1409
23 Wobbrock J O, Wilson A D, Li Y. 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. New York: ACM, 2007. 159–168
24 Carmona J M, Climent J. A performance evaluation of HMM and DTW for gesture recognition. In: Progress in

Pattern Recognition, Image Analysis, Computer Vision, and Applications. Berlin: Springer-Verlag, 2012. 236–243
25 Liu J Y, Zhong L, Wickramasuriya J, et al. uWave: accelerometer-based personalized gesture recognition and its applications. Pervasive Mob Comput, 2009, 5: 657–675
26 Reyes M, Dominguez G, Escalera S. Feature weighting in dynamic time warping for gesture recognition in depth data. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. Piscataway: IEEE, 2011. 1182–1188
27 Liu X, Mu Y, Zhang D, et al. Large-scale unsupervised hashing with shared structure learning. IEEE Trans Cybern, 2015, 45: 1811–1822
28 Liu X, Deng C, Lang B, et al. Query-adaptive reciprocal hash tables for nearest neighbor search. IEEE Trans Image Process, 2015, 25: 907–919
29 Chen M Y, AlRegib G, Juang B H. 6DMG: a new 6D motion gesture database. In: Proceedings of the 3rd Multimedia Systems Conference. New York: ACM, 2012. 83–88
30 Mitra S, Acharya T. Gesture recognition: a survey. IEEE Trans Syst Man Cybern Part C-Appl Rev, 2007, 37: 311–324
31 Ruiz J, Li Y, Lank E. User-defined motion gestures for mobile interaction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM, 2011. 197–206
32 Schlömer T, Poppinga B, Henze N, et al. Gesture recognition with a Wii controller. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction. New York: ACM, 2008. 11–14
33 Vatavu R D. Nomadic gestures: a technique for reusing gesture commands for frequent ambient interactions. J Ambient Intell Smart Environ, 2012, 4: 79–93
34 Pu Q F, Gupta S, Gollakota S, et al. Whole-home gesture recognition using wireless signals. In: Proceedings of the 19th Annual International Conference on Mobile Computing & Networking. New York: ACM, 2013. 27–38
35 van Seghbroeck G, Verstichel S, de Truck F, et al. WS-Gesture: a gesture-based state-aware control framework. In: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications. Piscataway: IEEE, 2010. 1–8
36 Zheng Y W, Sheng H, Zhang B C, et al. Weight-based sparse coding for multi-shot person re-identification. Sci China Inf Sci, 2015, 58: 100104
37 Hayashi E, Maas M, Hong J I. Wave to me: user identification using body lengths and natural gestures. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM, 2014. 3453–3462
38 Wang D X, Xiong Z H, Zhang M. An application oriented and shape feature based multi-touch gesture description and recognition method. Multimed Tools Appl, 2012, 58: 497–519
39 Chera C M, Tsai W T, Vatavu R D. Gesture ontology for informing Service-oriented architecture. In: Proceedings of IEEE International Symposium on Intelligent Control. Piscataway: IEEE, 2012. 1184–1189