Thesis

PersoSys: A User-defined Gestural Interface for Personalized Interaction

Submitted by

Ghazal Fahimi

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2015

Abstract

PersoSys: A User-defined Gestural Interface for Personalized Interaction

Recently, there has been an emergence of research projects pertaining to gesture-based interactions which have focused on procedures to design intuitive gestures. However, little work has been conducted in regard to discovering best-practices in the field of gesture customization by end-users—research which has the potential to vastly improve user experience beyond the potential of the predefined gestures. Gesture customization goes deeper into the gestural interface logic and makes it possible for users to map flexible tasks to their own defined gestures.

In this thesis, we describe the PersoSys system, a touchless, 3-dimensional gesture-based interaction framework, that makes use of an Intel Perceptual SDK camera to control the operations of a PC through hand gesture commands. The system allows users to define a personalized input gesture for a set of tasks, thereby enabling them to trigger those tasks by performing the mapped gestures. Furthermore, the user interface of the system is designed in an arrangement that offers customizations in a simple and convenient way, concealing much of the complexity from the user. In this thesis, we describe the hardware and software implementation of PersoSys, and several use cases designed to bring more convenient interaction to users. We also examined the effectiveness of our system by conducting an experiment that obtains qualitative and quantitative feedback from users. The results imply that users are pleased with the configuration process and automatically executing the long sequences of tasks by performing their desired gestures.

Table of Contents

List of Figures

CHAPTER 1

Introduction

The notion of developing application user interfaces with the goal of bringing appealing interactions and matching the abilities of users has long been one of the aims of human-computer interaction researchers. In innovative software environments, personalization and customization of the system's behavior provide more intuitive and natural interactions, improve the usability of software for a wide range of users, and bring a more satisfactory user experience than presently designed systems [14].

In the domain of human computer interaction (HCI), novel technical approaches have been introduced that create more intelligent software environments and better support user interaction. Most of these technical methods have a capability of being customized and modified by end-users, which supports a more intuitive, natural, effective, and memorable interface. One of these approaches is to allow users to control system using hand gestures, a class of gestures based on the movement of the human arm and hand in 3-dimensionals, which promotes intuitive and natural user interaction [18]. Moreover, in the domain of *end-user programming*—techniques that allow users who are not skilled software developers to program computers—software engineers have proposed techniques that automate the execution of tasks or modify behaviors of software [6].

Recently, a wide range of studies has been conducted focusing on the afore mentioned techniques to support more natural and usable computer interactions. However, there are still opportunities to bring software closer to human's interests and demands using these approaches considering that people have different opinions on evaluating software in terms of interactivity and usability. These opportunities lead the author of this study to benefit from these techniques and apply personalization and customization in software [29].

In this thesis we present *PersoSys*, a system that supports the personalization of a computer's behavior. This system acts as an intermediary bridge between a user and their computer by enabling a user to define their own hand gestures for the sets of flexible computing tasks. PersoSys provides an informative user interface that makes the creation of gestures for tasks easy and convenient. Furthermore, it allows users to actively define new mappings between gestures and tasks as time persists. This work will contribute to the design and development of a new generation of intuitive and personalized user interfaces, as well as making the computer interactions more natural, convenient, and effortless.

Our work brings together the ideas and approaches from many previous accomplished systems and studies. For instance, we include the aspect of defining personalized sets of gestures to control medical systems from the work accomplished by Bigdelou et al. [2], the concept of customization by enabling users to design their own personal gestures obtained from Oh et al. [17], the technique of automating the GUI interaction from Sikuli [29], and the experience of making the computer interaction more intelligent by using progressed gesture recognition cameras of WorldKit [27].

The rest of this thesis is organized as follows. First, we will provide an overview of related work followed by a description of the design goals of PersoSys and its main components. We then turn to implementation details, discussing the architecture of the system, the hardware used, and the techniques in recording the user-defined gestures and assigned tasks. Afterward we provide several example applications and discuss how PersoSys can be beneficial in different situations and make interactions more appealing for users, specifically for people with particular disabilities. Next, we present an experiment to conduct qualitative and quantitative feedback regarding the preferences of the users on designing their own gestures for tasks, and the system's performance on achieving this goal. Lastly, we will discuss the findings of our study, limitations, and future work.

CHAPTER 2

Related Work

Relevant prior work includes studies related to user preferences of gestural customization and its benefits over pre-defined gestures, systems investigating using custom interfaces, an the contribution of end-user programming to personalized computing.

2.1. User Preferences of Gesture Customizations

Interacting with hand gestures has shown to be more natural and faster than using mechanical devices, such as a keyboard or mouses [18]. Recently, user interface designers have employed hand gestures as a method of interaction in different devices such as large displays [25, 16], desktop computers [3, 12], televisions [23, 24], keyboards [12], and mobile phones [10].

Despite the wide use of gestural interaction, the challenge to design natural hand gestures for appropriate computer operations still remains. Hand gestures are typically designed by system designers. Despite being proficient in design, this approach is known to create arbitrary, unintuitive [9], and unnatural interactions. Therefore, HCI researchers tried to overcome these flaws by designing elicitation studies that obtain intuitive gestures for computing tasks directly from end-users. The result of these studies is a set of user-defined gestures. These gestures are known to be more appropriate, intuitive, and natural gestures set than those designed by designers [26].

Wobbrock et al. [26] was one of the first to examine user-defined gesture sets. As part of their work, they presented the consequences of 27 tasks on a digital tabletop to participants, and asked them to design a gesture to perform each task. The result of the study was a user-defined gesture sets that provided good candidates to bring interactive surface computing closer to the behavior of users. In follow-up work, the authors demonstrated that many participants prefered the user-defined gesture set over a set of gestures created by system designers [15]. Wobbrock's experimental design has inspired a number of studies to extract user-defined gestures by conducting similar elicitation studies [26]. These include a study on gestures for data transfer and communication between iPads and other devices [8], eliciting 3-dimensional user-defined motion gestures for smart phone interaction [21], user-defined gesture set for augmented reality (AR) applications [20], and interactions with 3-dimensional objects from touchscreen inputs [4].

It was assumed that previous elicitation studies were bringing natural and intuitive computer interaction. However, HCI researchers have recently conducted some studies that represented the contrary inference. These experiments demonstrated that the obtained gesture sets from the elicitation studies in fact are not as intuitive for as wide range of users as they could be [13, 17]. They also presented that the users are still forced to follow some undesirable system designed gesture sets even if they are extracted from user-defined gesture methods. These discoveries motivated many researchers to introduce the customization of gesture interfaces approach that offers more usable interaction.

One of these studies that addressed the flaws in elicitation studies was established by Maunery et al. [13]. They investigated cultural differences of gestures performed by 340 participants in nine countries across Europe, Asia and North America. Overall, they observed high agreement of user-defined gestures across the countries. However, they did notice that the Chinese participants prefer to design more symbolic gestures, such as letters, than participants from other countries. Furthermore, they discovered that there was lower *agreement scores*—which are computed from dividing identical user-defined gestures by total user-defined gestures for each task [26]—on actions that are more symbolic in nature. The finding of their research is consistent to the motivation of our work that personalization of a gestural interface is an important approach toward providing intuitive interaction for a broad range of users.

Moreover, most of the elicitation studies that we mentioned before were focused on eliciting the gestures for the tasks that specific gestures previously had assigned to them. For example, scroll down and up are two common tasks that some gestures assigned to them. However, researchers have still investigated these actions in their studies and they gain high agreement scores related to them. The reason of high agreement score for these tasks according to Oh et al. [17] is that users are familiar to these types of common tasks and their mapped gestures. Therefore, users are biased—meaning that they have a tendency to think in certain ways that is caused by doing repetitive actions—to pre-existing knowledge leading them to design the same gestures again. However, the real challenge is finding a unique and natural gesture for every novel task that is new to each user. Meeting this requirment is impossible in elicitation study since the participants cannot design gesture for every possible computing tasks. To overcome this challenge, Oh et. al [17] suggested gesture customization by enabling users to design their own gestures for tasks.

Another issue of elicitations studies addressed by Oh et al. [17] who demonstrated the inapplicability of some elicitation studies' results. In one set of the experiments implemented to determine the type of preferred user-defined gestures, the participants were required to create new gestures for a general, open-ended use context. During the experiment it was observed that some of the participants had a misconception about the recognizer's ability, thereby leading them to design gestures that could not be properly classified. Conversely, this same misconception led some of the participants in the opposite direction of excessively constraining the type of gestures they created. In fact, the observation of the study demonstrates that since in previous gesture eliciting research there was no quality evaluation of an individual user's gestures based on the recognizer's ability, the results may not be valid or applicable to real devices supporting gesture interaction.

2.2. Customizable Gestural Interfaces

The arguments of previous studies, which focused on providing customization capability for users, convinced us to maintain a gestural user interface which could be personalized to only one user with regard to their special situations, interests, and abilities. One particular example being people with physical disabilities who require unique customization significantly different from that of unimpaired users.

Our work is not the first study targeting customization of gestures in devices; various systems have previously conducted gesture personalization capability [2, 14, 11]. Bigdelou et al. [2] established a gesture-based interaction solution for the operating room. They presented an interface that enables surgeons to define a personalized set of gestures that best fits particular requirements and showed that the proposed gesture-based interface provides good usability factors for applicability. Liu et al. [11] developed uWave for interacting with mobile or electronic devices using on personalized gestures and physical manipulations. In the study, Liu et al. mainly focused on accuracy assessment of their proposed gesture recognition algorithm, but the main features of uWave presented a technology that facilitates personalized gesture recognition. Furthermore, Merill et al. [14] built up a musical control application called FlexiGesture that allows a user to define input gestures, and modify the sounds separately and finally assigning them together.

The primary goal of all the mentioned work was designing gestural interfaces that could be configured and personalized according to a users' desire, but none of the studies focused on customizing gestures that could be associated with common software in PCs and Laptops, a goal of our work.

2.3. End-user Programming

End-user programming is best known for improving the end-user interaction by enabling users to define their own programs. According to Hoc et. al. [6], "programming" is the process of transforming a desired computer action from a user's mind into a representation that can be understood by a computer. Also, "end-user programming" is defined as "people who write programs, but not as their primary job function, they write programs in support of achieving their main goal, which is something else, such as doing office work, scientific research, entertainment, or engineering" [22]. Today, a significant part of research and tools are written for supporting end-user programming and improving software control, automation, and direct manipulation. Macro programming and scripting are the techniques in direct manipulation interfaces supporting automation of repetitive tasks and representation of low-level, tedious activities [30].

Recently, Yeh et. al. [29] developed Sikuli, a visual approach to search and automation of graphical user interfaces using screenshots. It provides a visual scripting API for automating GUI interactions to direct mouse and keyboard events. Zettlemoyer et. al. [30] established VisMap that supports the control of an application through its graphical user interface. Also, they presented VisScript which generates a set of scripting commands related to mouse actions based on the output of VisMap. It provides users a simple facility for running visual scripts.

The distinction between prior work and the work presented in this thesis is that the main focus of prior work was maintaining task automation by using the screenshots of the manipulated objects in the tasks, as well as requiring users to write certain programs to configure tasks automation. Our goal is to enable users to automate more general commands and actions by merely executing them with hand gestures. However, their main concepts in automation of GUI commands had inspired us to build PersoSys, software that is capable of end-user programming which enables users to automate the repetitive and tedious task in order to be performed more quickly and reliably. Also, by mapping each of the task to a self-defined gesture, our system will help the user to save more time, engage in more smart computing environment and reduce the incidence of errors caused by manual repetition of tasks.

CHAPTER 3

PersoSys

3.1. Design Goals

We developed an interface called PersoSys which attempts to address challenges by fulfilling three basic design goals: improving the convenience of common user interactions, being manageable and intuitive for a broad range of users, and instantly accessible in multiple environments. PersoSys acts as an intermediary between a system controller and a computing task to be controlled, which we call "users" and "tasks" respectively.

Our first goal developing PersoSys was to make a wide range of human-computer interactions more convenient. In this context, convenience implies not only that users should feel no physical discomfort, but also that the action should feel natural, be repeatable, and be inherently representative of the command they are targeting to perform. To achieve this desired convenience, PersoSys is operating as an intermediary between the user and computer. It provides more convenient human computer interactions by enabling users to control and perform tasks intuitively, easily, and quickly. Users can personalize hand gestures to operate the system, as well as a sequence of computing tasks triggered by the gesture. Furthermore, it doesn't necessarily require users to finally configure the system nor define the desired parts of a hand gesture and tasks before using the system. Indeed, it allows them to actively customize the behavior of the software and modify previous configurations in a manner reflecting their changing demands, thoughts, and desires as time goes on.

Our second goal was to design appealing user-definable software that is easy to use for a broad range of users independent of their technological expertise. PersoSys must not only make interactions with a computer effortless and simple but also be understandable and easily manageable for users. Our first goal already mentioned that our system should make interaction with a computer easy and practical, however it's important to emphasize that ease of use of the system is of utmost importance, and must require minimum user's knowledge and effort to be employed. We meet this goal by providing a user-friendly, simple, straightforward, and intuitive user interface that guides users through defining new gestures for tasks as well as managing the system easily.

The third design goal is to make PersoSys portable and easy to set up in different environments. PersoSys must be reusable and ubiquitous to allow it to be accessed instantly when and where it is needed. The system should be applicable to and usable in different environments where PCs and Laptops are present. In order to meet this goal, we employ the small and portable Intel Perceptual computing SDK sensor.

As mentioned in the related work section, motion gestures and end-user programming are two approaches in building a convenient and customizable software environment as well as intuitive human-computer interaction. We try to create a mapping capability between the user defined motion gestures and the sequences of tasks that users intend to execute by performing the defined gesture. PersoSys constructs a simple, intuitive and interactive bridge from a user's intention to task execution and completion.

3.2. PersoSys Features

Hand gestures are known to be more natural, simpler and faster to perform than a keyboard or mouse for controlling computers [28]. In addition, automating computing actions that are repetitive, tedious, or consist of long sequences of tasks is a big step toward creating intelligent software environments. We developed PersoSys to allow users to have more convenient interactions with computers mainly based on the techniques we have just mentioned.

PersoSys learns hand gestures from examples provided by a user who can freely define the movements and particular sets of computer tasks that fit each other best. Gestures are perceived by using an Intel Perceptual Computing SDK camera. This sensor is capable of capturing the (x, y, z) coordinates of fingers and palms which makes the hand gesture detection very precise.

PersoSys consists of two modes: training and service. In the training mode, there are two phases in which the user defines a pair of a hand gesture and a sequence of tasks. In the first phase, the user can design a hand gesture using one or two hands by performing it in front of the camera. Meanwhile, the interface of PersoSys displays the depth video and recognized hand's points to help the user locate their hand in an appropriate position in front of the camera. PersoSys asks the user to perform the gesture once, since it requires a single sample to train each gesture pattern. As the user is performing the gesture, PersoSys will capture the points data of the hand and record it to the computer's hard disk. The user can select the recording option to record a video of the performed gesture, so that she can watch it later to ensure correctness. In the second phase, PersoSys expects the user to define the set of computer tasks she wants to be executed by the recently designed gesture. The user performs the sequence of tasks using the mouse and keyboard while simultaneously PersoSys records these actions and saves this information. Just like the gesture definition phase, a user can watch the recorded tasks later to verify its validity.

In the service mode, the user can execute the defined tasks by performing their corresponding mapped gestures. PersoSys runs continuously as a service program, thereby, allowing the user to perform defined hand gestures which consequently performs the corresponding tasks. However, the user can control the service by turning it off or on using a menu embedded in a system tray icon.

Consider a scenario in which a user whose primary job is translating books is interested in automating certain tasks. For example, to start a new book translation she needs to open the internet browser and the dictionary website, a specific E-book that she is going to translate from, and a particular text file she is writing the translation to. She may also be interested in opening up her E-mail account, a couple of websites related to translation-editing, or playing a list of music on her computer. Instead of performing each of these long, tedious, and repetitive tasks manually, PersoSys helps this translator to automatically execute all of these tasks by recording a hand gesture she designs and then mapping the tasks to the gesture. In this case, the user defines a big hand (high five) gesture at the training session and performs all of the desired tasks. PersoSys watches and records all of these commands. At the next run the PersoSys service will consider the defined map and when the user shows a big hand gesture to the camera, the mapped tasks will be executed.

PersoSys meets the first design goal, bringing a convenient interaction, by detecting the user-defined and intuitive hand gestures and consequently automating the repetitive and tedious mapped tasks. It supports user personalization of the set of gestures and tasks she is interested in and adds more configurations as time goes on. It also matches the second goal of ease of use. Working with PersoSys does not require users to be computer experts since it provides a user-friendly and informative interface that helps them to install and define their own configuration, simply and effortless (Figure 3.1). Also, it satisfies the third goal, ubiquitousness, by employing a small and portable camera that could be set up on the computers easily anywhere.

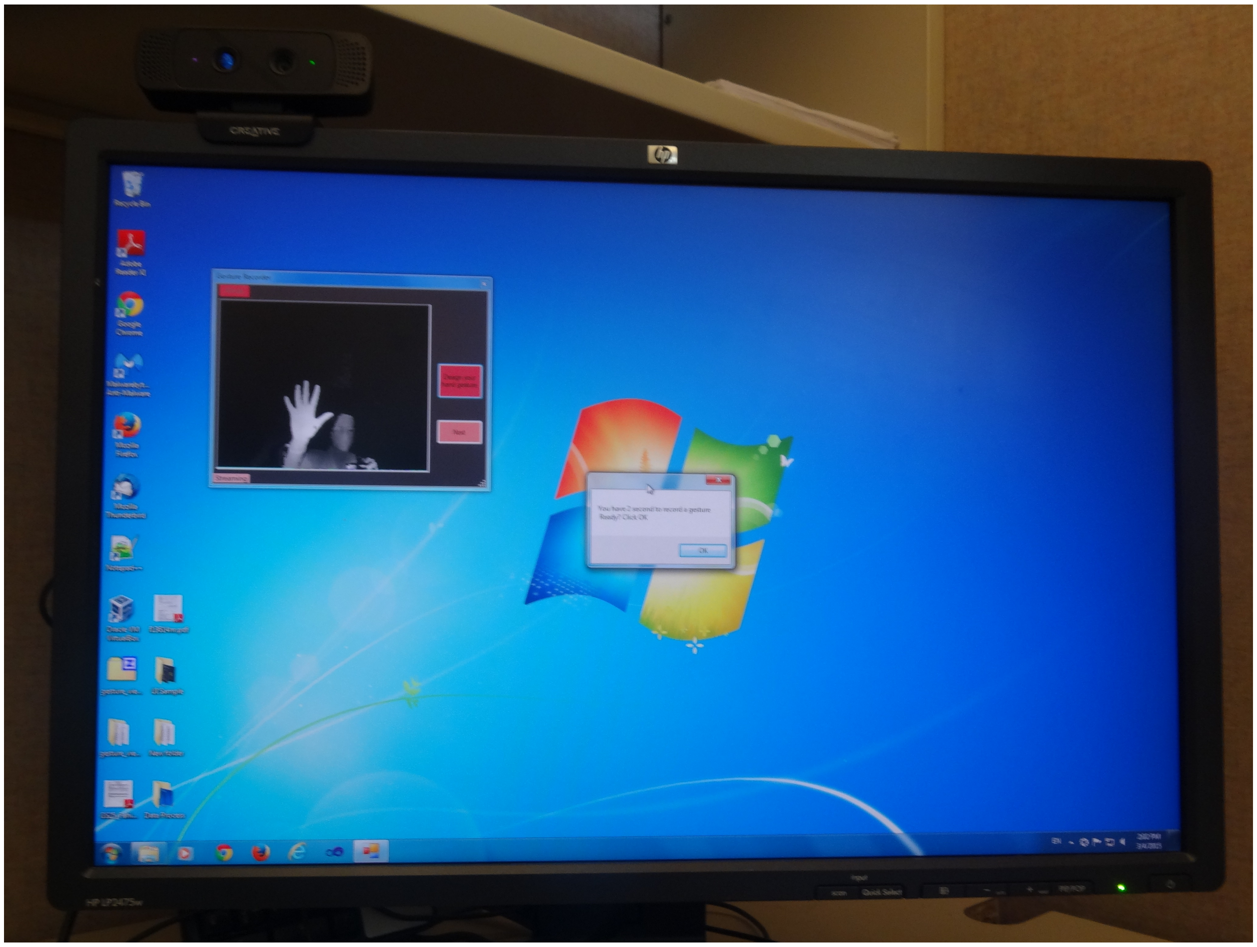


FIGURE 3.1. The PersoSys setup. Using the Intel SDK camera, PersoSys eanbles users to define hand gestures for tasks.

The next section describes the architecture and implementation of PersoSys in detail and explains how each component of PersoSys meets up the design goals we discussed.

CHAPTER 4

System Implementation

4.1. Architecture Overview

In this section, we give a brief description about the architecture of PersoSys and discuss how it meets the design goals mentioned in the last section. Based on the defined goals and the two introduced approaches, user-defined motion gestures and end-user programming, we design and demonstrate the main architectural segments of PersoSys. The software includes four main components: sensors, a recorder, gesture recognition, and effector.

Sensor component: The *sensor* component itself consists of two parts. The first part processes gesture input from the user. This includes receiving the information of a user's hand motion gestures after the camera has perceived the user and their hands. This component communicates with and receives data from a sensor device named Intel Perceptual Computing SDK that is portable and easy to configure in different places. Thus, it supports the third design goal which is ubiquity and portability.

The second part of the *sensor* component monitors the mouse and keyboard actions. During tasks recording, this portion of the sensor captures every mouse and keyboard action by using handlers that are responsible for controlling the input events from keyboard and mouse. In these ways, both parts of the *sensor* component play an intermediary role in achieving both the first and second design goals by capturing personalized command information from the user.

Recorder component: The *recorder* component is responsible for receiving new information regarding hand motion gestures and tasks from the *sensor* component and saving it onto the system's hard drive. This part plays an important role in meeting the first design goal which was to provide the software with customization through personalized interaction, including both defining new gestures and assigning sequences of tasks to gestures, while simultaneously making the interaction convenient. In addition, the user is not required to demonstrate all of their gestures in the first usage of software as PersoSys allows the user to actively define new motion gestures every time.

Gesture recognition (GR) component: The *GR* component enables the software to recognize the gestures previously defined by the user. The second set of goals necessitates that this component be implemented in an arrangement capable of detecting any number and any kind of hand motion gestures performed by a user who may not have any knowledge or experience in the software.

Effector Component: The *effector* component allows the system to execute the sequence of tasks which are mapped to recognized gestures. This component also meets the second sets of goals, accomplished by working closely with the *GR component*.

Figure 4.1 illustrates the general architecture of PersoSys and all the components it consists of, as well as the communication and association between these components.

4.2. Sensor component

This component is responsible for capturing the data inputs produced by a hand's point's location as perceived by the Intel camera as well as keyboard and mouse actions. Due to the differing types of input, we divided the *sensor* component into two separate segments which we named the *motion gesture sensor (MGS)* and the *keyboard and mouse gesture sensor (KMS)* which perform different roles.

The *MGS* obtains the data information of the hands' points which are detected by the Intel Perceptual SDK camera. The Intel Perceptual Computing SDK is a camera that

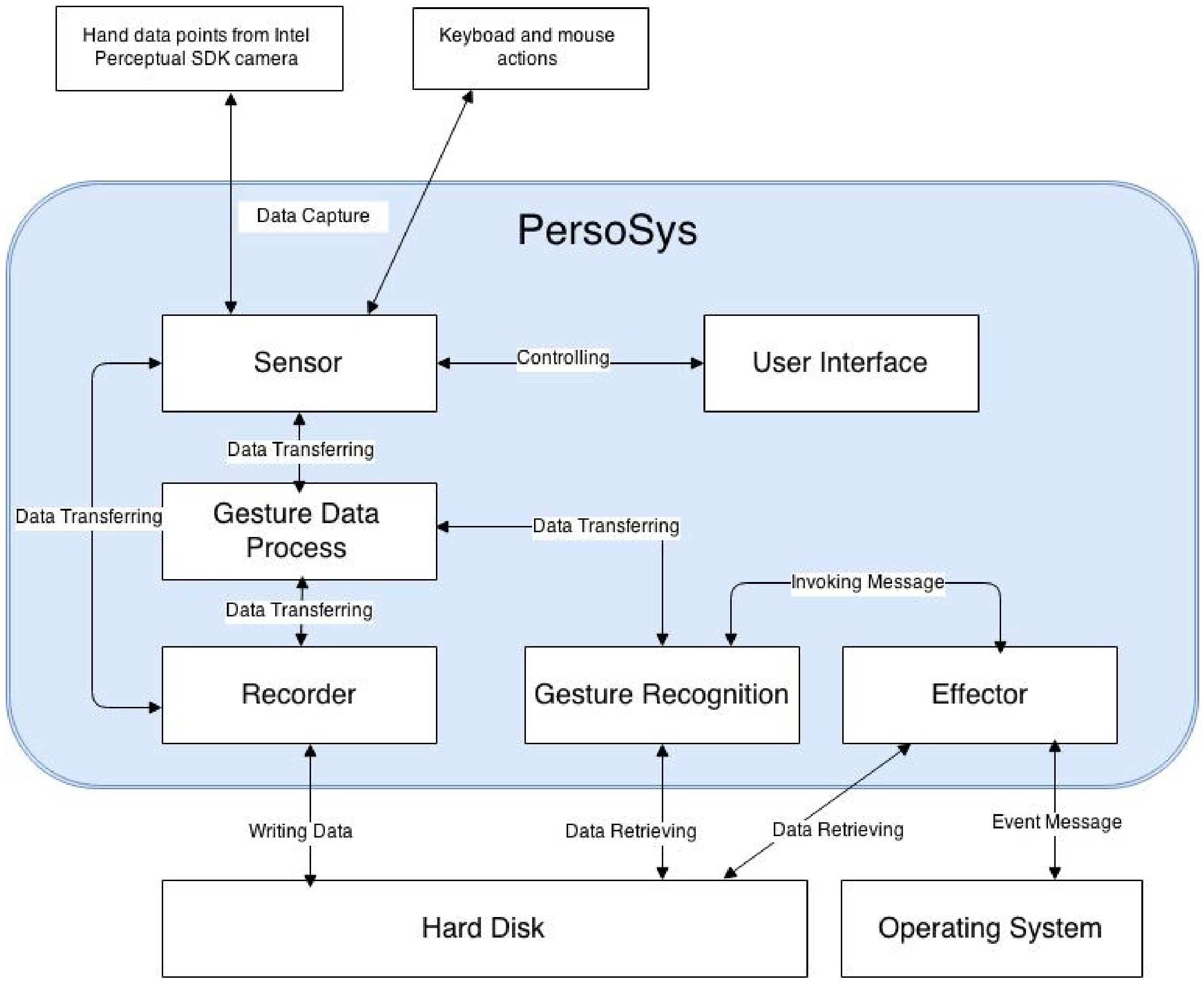


FIGURE 4.1. PersoSys general architecture and communication.

provides RGB and depth images. This capability works from 0.2 - 1.2m from the camera and extracts 3D positions of the fingertip, palm, and forearm locations. The Intel camera is connected to our system using the *MGS* segment. This part is responsible for obtaining the perceived gesture data and transferring this information to the *Gesture Data Process (GDP)* component. It is worth mentioning that in the context of our software usage, the computer user is highly engaged in computer operations that involve keyboard and mouse manipulation using their hands and fingers. Therefore, although a wide range of body movements can be implied as gestures, we are focusing on subsets of gestures that are shaped and performed by hands and fingers. Since we consider only the motion gestures performed using the hands, we used the Intel camera which supports finger and palm detection more precisely than other comparable cameras.

PersoSys must be able to record and save the information associated with desired computer tasks to perform them later. The tasks in this context are those computing functions that are the result of input from a keyboard and mouse. The *KMS* is in charge of capturing the keyboard and mouse actions. It installs some handlers to process the events coming from keyboard and mouse actions and obtain their associated information in the task recording session. *KMS* encapsulates the mouse action in a *MouseEventArgs* object that includes the (x, y) coordinates and the click button properties. Also, it encapsulates the key action in a *KeyEventArgs* object that includes a feature about the type of pressed key. *KMS* pushes the actions on a list and passes it to the *Record* component for writing and saving the associated data onto the hard disk.

4.3. Gesture Data Process Component

Intel camera perceives 11 points for each hand. By trying various approaches, we discovered that 8 among those 11 points led to a more accurate recognition. Therefore, we removed three specific points. One of these three points with the `LABEL_ANY` label states that the Intel camera is looking for any gesture. This point is utilized for the gesture recognition application of the Intel camera and it actually doesn't provide any information regarding the position of the hand's points. Figure 4.2-A illustrates all the 10 points provided by the Intel SDK while Figure 4.2-B illustrates the 8 points used by our system. The red circled points illustrate those points that we removed from our set of points. The point with the `LABEL_ANY` label is not shown on the Figure 4.2.

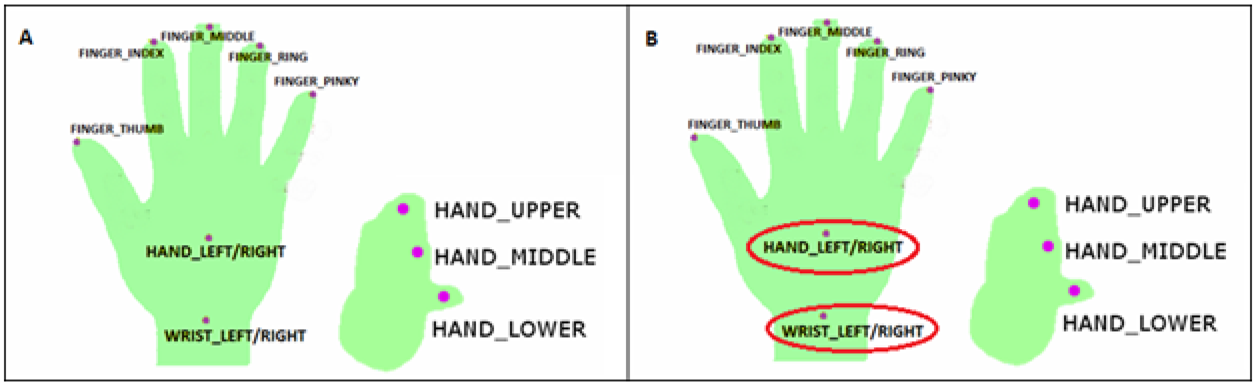


FIGURE 4.2. The labels and positions of the hand's 10 points that Intel camera provides (A). The hand's 8 points we used in gesture recognition excluding the red circled points (B).

The hand motion gestures data captured by sensors may actually vary widely from the one originally performed by the user in terms of the hand's position and coordinates. To make the data more robust against the position of the gesture, the *gesture data process (GDP)* component implements a transformation on perceived hand location. Because the `Hand_Middle` labeled point of a hand has less variance in terms of its position than hand's other points, we subtract the `Hand_Middle` point value from all other point's values. We arrived at this approach by exploring other point transformations and measuring the recognition accuracy. We concluded that the `Hand_Middle` point gives the most accuracy.

In addition, PersoSys supports both static and dynamic hand gestures recognition. Static and dynamic gestures differ in their shapes and movements over time. Static gestures have the same shape and do not vary in time, e.g. pointing. Dynamic gestures are marked by changing the shape and moving in time, e.g. waving or drawing letters in the air. In order to make the data robust against the movement and speed of a gesture and to detect both static and dynamic gestures, we designed a data process algorithm that is applied to data before passing it to the recorder and *GR* components. The hand's points movement is the determining feature that differentiates between dynamic and static gestures. The

GDP component obtains the speed of each hand's points movement in each frame, which is calculated by subtracting the values of their previous corresponding (x, y, z) vector position from the current position. We used the result to obtain the (s_x, s_y, s_z) speed vector for each point. The frame rate of the Intel camera is stable during recording. Therefore, we don't divide the calculated difference value by time, as dividing all data values by a static value doesn't have a useful effect on dynamic time warping classification, the gesture recognition algorithm used by PersoSys. The obtained speed vector is used to distinguish between static and dynamic gestures.

4.4. Recorder Component

In training mode, the *recorder* component receives information regarding new user-defined gestures and tasks from the *GDP* component and *KMS* segment. The *recorder* component then writes the new gesture information in an (x, y, z, s_x, s_y, s_z) format in a blank file using the .gest filename extension (an extension created for our system). As we mentioned, the mouse actions are encapsulated in a *MouseEventArgs* object and the key actions are encapsulated in a *KeyEventArgs*. The *recorder* component writes each new set of task data information in a format of a list including each action's object types and properties in a blank file using the .tsk filename extension.

The *recorder* component tracks the number of defined gesture and task pairs. For each new pair, it increments this number and uses it for naming the .gest and .tsk files the same name which is the number of the new pair. These files are in a PersoSys directory located in the user's home directory.

4.5. Gesture Recognition Component

The *GR* component is called to recognize a perceived gesture (test data) during the service mode. It has an association with the *GDP* component and stored data. After the *GDP* component converts the test data to a more meaningful and robust set of features, it passes that data to the *GR* component. The *GR* component's output is the classification of the received test data.

The *GR* component mainly consists of a gesture recognition method. PersoSys must recognize the hand motion gestures that the user has already defined. Since there could be a considerable number of user-defined gestures that are saved in the computer's hard disk, the recognizer algorithm ought to be accurate in classifying the perceived gestures in an adequately short time. The recognizer must discard the gestures that are not in the set of defined gestures, minimizing false positives. When a user performs a registered gesture, it is not uncommon for the gesture's speed and size to differ from the saved data. Therefore, the classifier should be speed and size indifferent [19]. Even more so than previous parameters, the gesture recognizer should be highly accurate and precise in its decision making otherwise the user will be frustrated about the system performance.

All of these arguments motivated us to employ a gesture recognition algorithm with fast and accurate performance. We implemented the dynamic time warping (DTW) which is an algorithm for measuring the similarity between each defined hand motion gestures—which already had been registered to systems—and a test hand motion gesture—a performed gesture by the user that is intended to be the same as one of the registered gestures—which may vary in size or speed. DTW just needs one training sample for each class of gesture. Through this feature, the system provides convenience in defining a gesture to system, training mode, and it does not require the user to define the new gesture more than one time to system. Also, it decreases the amount of calculations or memory space usages.

After the DTW algorithm identifies the class of the test gestures there is a threshold that minimizes false positives in order to obtain more accurate results. The threshold is a specific number and responsible for filtering the results of the DTW algorithm. If the number of optimal distance is greater than the threshold, then it means that the test gesture is not actually the same as the classified gesture. In this case, the recognizer will discard the result and classify the next test hand motion gesture. The number we assigned to the threshold did not come from a specific formula; it was the result of trying various numbers, measuring the result, and comparing the related accuracies.

Since our focus was improving the HCI features of PersoSys more than its machine learning features, we put limited effort into the *GR* component. In the future we would like to experiment with the accuracy of various motion gestures classifiers and threshold numbers over a wide range of participants.

4.6. Effector Component

After the user performs a gesture and the *GR* component detects the class of the gesture, the *GR* component calls the effector component. Since both the defined gesture and the mapped tasks files are saved with the same numbers, the effector component receives the assigned number of the detected gesture from the *GR* component and opens up the associated .tsk file.

The effector is then responsible for reading the information concerning keyboard and mouse actions from that specific *number.tsk* file and encapsulating the data. As we discussed in the *recorder* component section, the mouse and key actions are in the form of *MouseEventArgs* and *KeyEventArgs* objects. The recorder then writes the properties of the objects in a file. When the effector is reading a file, it will populate the data back into *MouseEventArgs* and *KeyEventArgs* objects appropriately. Finally, effector passes mouse and key events to the operating system to perform the desired action in an automatic fashion.

4.7. Graphical User Interface Component

The graphical user interface (GUI) of PersoSys is responsible for guiding the user in defining the set of hand motion gestures and tasks in a training session as well as controlling the system during service sessions. PersoSys has two different GUIs, one for training and one for service mode.

4.7.1. GUI in Training Mode.

Figure 4.3 illustrates the GUI during training session. It is designed in a simple and friendly style and consists of a few widgets including panels, menus, and buttons. The *Gesture Recorder* window helps the user define a new hand motion gesture. In the upper left section of the window, there is a *mode* menu. It consists of three options: *Live*, *Record*, and *Playback*. *Live* is the ordinary mode that records the hand motion gesture in terms of the location of its data points. *Record* records a depth video of the gesture as well as recording the gesture data points. Finally, the *Playback* option plays the video of the gestures that have already been recorded in *Record* mode. The *Gesture Recorder* window also consists of a panel displaying the user's depth videos captured by the Intel camera and the set of recognized hand points. It acts as a mirror, helping the user to locate their hand in an appropriate position, thereby seeing how their hand motion gesture was perceived by camera. This panel starts to work after the user hits the *Design your hand gesture* button.

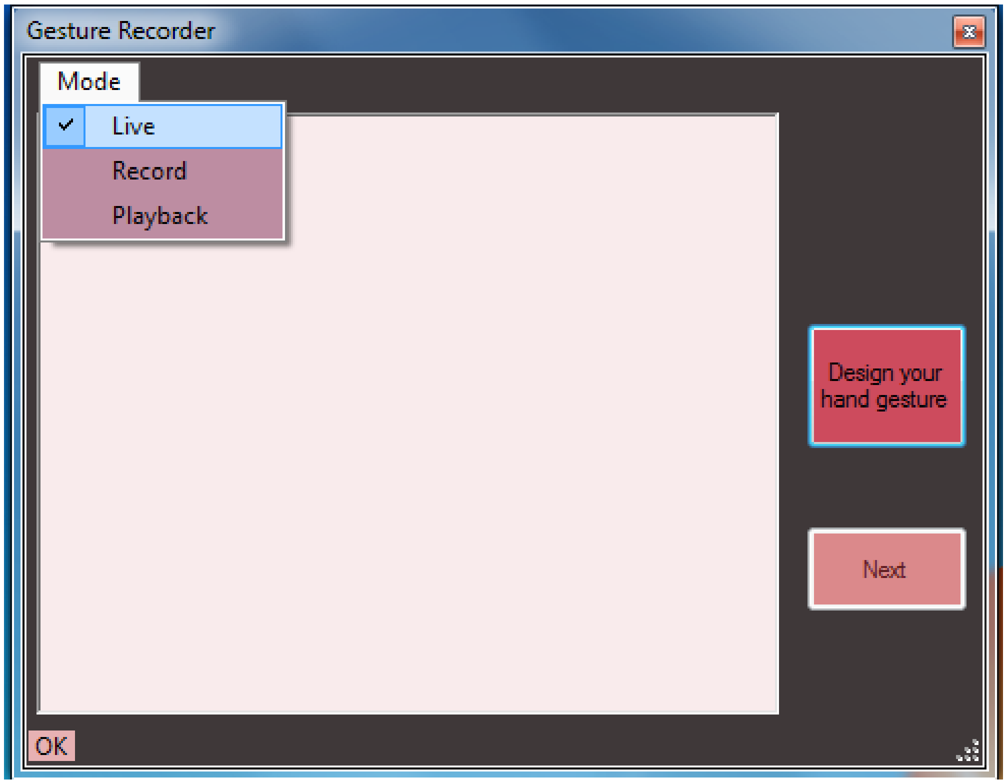
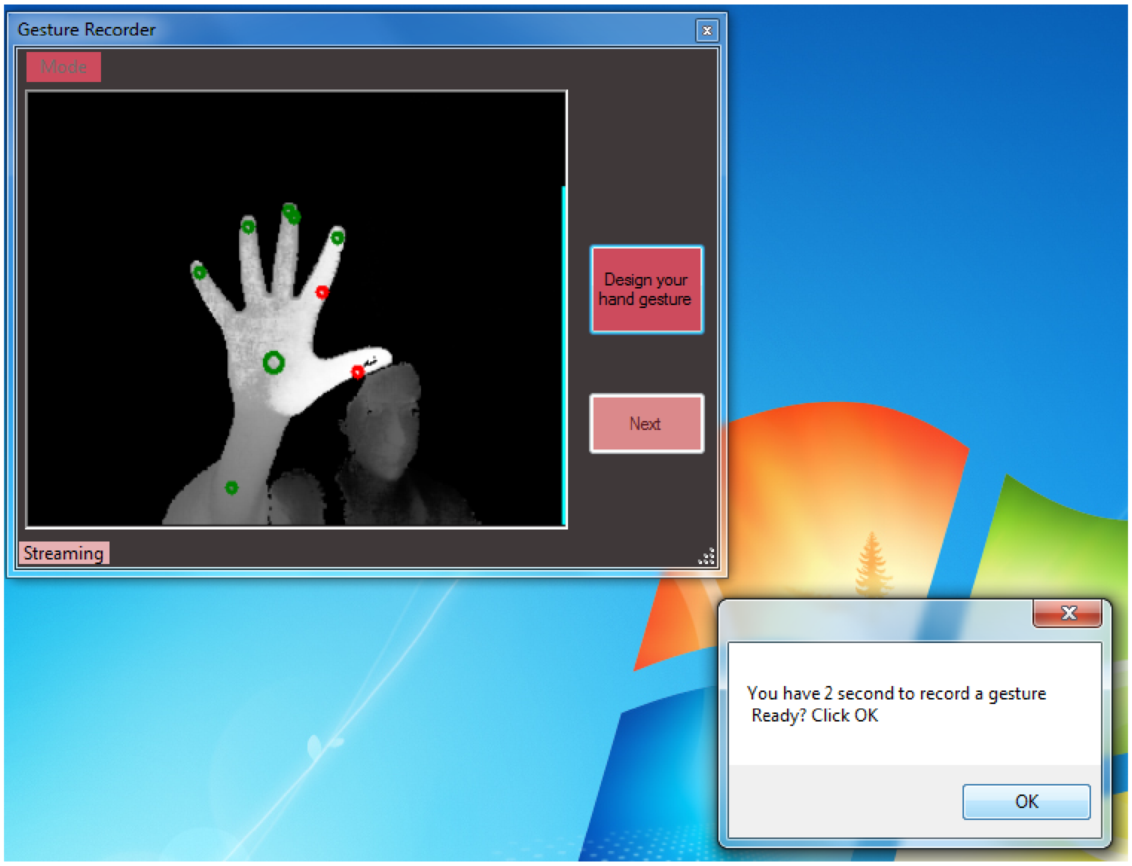


FIGURE 4.3. The graphical user interface of PersoSys in the start of training session. The *Gesture Recorder* window is responsible for helping the user to define new gestures.

In both *Live* and *Record* modes, when the user hits the *Design your hand gesture* button, a message shows up informing the user that the recording session will continue for two seconds. The left panel also starts representing the depth video of the user (Figure 4.4). When the user has sets up their hands and is ready to record a gesture, she can hit the *Ok* button in the message box and perform the gesture. After 2 seconds the recording will stop and the *Next* button will be enabled. In *Record* mode, the user is asked at the beginning of the gesture recording session to specify a name and location to save the gesture video file.

After completing the *Gesture Recorder* session and pressing the *Next* button, the *Task Recorder* window appears and asks the user to record sequence of tasks that is going to be performed by the recently defined gesture. The user hits the *Record your tasks* button and she starts executing the tasks. After performing the tasks, she may hit the *Replay the*



You have 2 second to record a gesture
Ready? Click OK

FIGURE 4.4. When the user clicks the *Design your hand gesture* button a message shows up informing the user that the recording session will continue for two seconds.

recorded tasks button to verify the tasks or the *Finish* button to close the training session (Figure 4.5). In order to define a new configuration, the user runs the training session program again.

4.7.2. GUI IN SERVICE MODE. Figure 4.6 illustrates the GUI in the service session period. There is a system tray icon that is located in the windows taskbar (usually at the bottom next to the clock) which functions as an interface for monitoring the service and easy access to system functions. When the computer is turned on, the system starts running to detect the user's gestures. At this stage the icon is red and it displays a *Tracking* status meaning the system is perceiving hand motion gestures. If the user performs a gesture that

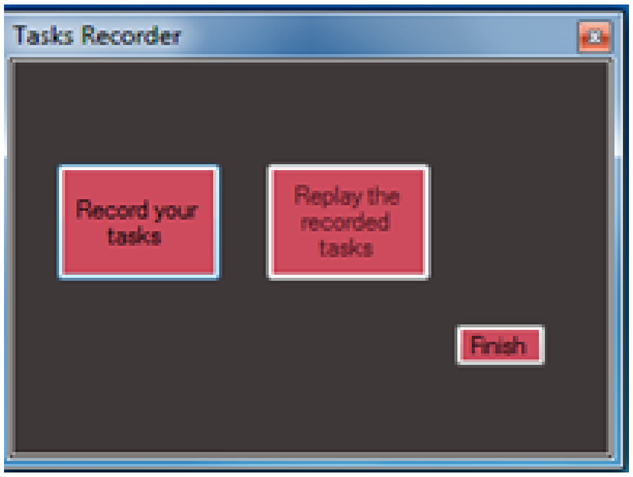


FIGURE 4.5. The *Tasks Recorder* window guides to user to record a sets of tasks.

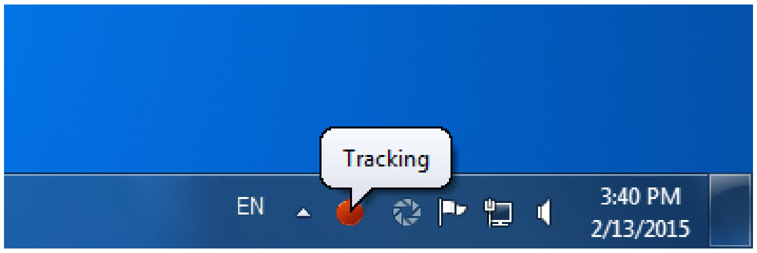


FIGURE 4.6. The GUI that displays in the service session. The system tray icon is red and *Tracking* text means the system perceives the hand motion gestures.

PersoSys detects, then the icon will turn to a green color and show *"A gesture is detected"* (Figure 4.7). However, if PersoSys does not recognize the gesture, it will stay red. The user can turn the service off by right clicking on the icon and choosing the *"Stop PersoSys Service"* option from the menu (Figure 4.8). The service will start running again if the user selects the *"Start PersoSys Service"* option from menu (Figure 4.9).

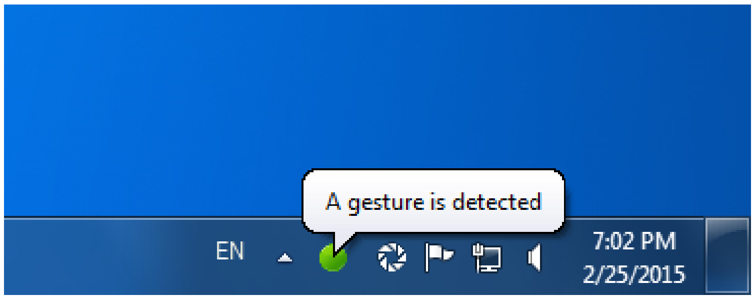


FIGURE 4.7. PersoSys recognizes a user-defined hand motion gesture.

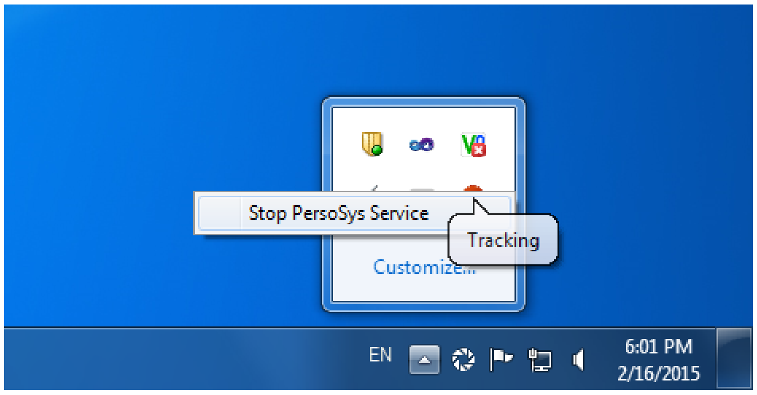


FIGURE 4.8. User can stop the PersoSys service by clicking on *Stop PersoSys Service* button.

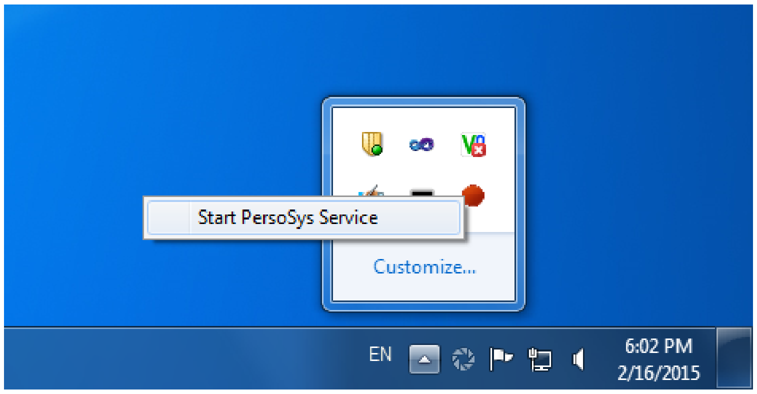


FIGURE 4.9. User can start the PersoSys service by clicking on *Start PersoSys Service* button.

CHAPTER 5

Use Cases

To illustrate the utility and capability of our software, we describe several examples of using PersoSys in different situations.

5.1. Blind Computer Users

Providing positive and productive computer interactions for blind and visually impaired people presents a challenge for the designers of computing systems. While gestural interfaces are now present across a wide range of computing devices, most of them provide few or no approachability features for blind people consequently leaving them largely unusable [7]. PersoSys provides a means to automate computing tasks that are tedious or difficult to perform by keyboard or mouse, especially for blind people. Although the main goal of developing PersoSys is not to satisfy the visually impaired users' requirements and in some aspects it lacks necessary applications for blind people, PersoSys brings a novel and practical approach toward effective human computer interaction for blind people. Instead of having to perform repetitive tasks such as launching and configuring programs as well as browsing to frequently accessed websites using a keyboard and mouse, PersoSys enables blind users to assign a favorite hand motion gesture to such tasks and automatically execute the commands which lead to speed up and help to facilitate the interaction.

5.2. PC-users with Parkinson's disease

Parkinson's disease (PD) is a highly prevalent and disabling condition and most of the PC users with PD have major challenges and serious problems related to manipulation of the keyboard and mouse [5]. Therefore, improving their computing experience is important.

We propose an alternative solution by replacing these problematic peripheral devices with PersoSys and allowing PD users to bypass a series of frustrating tasks by performing gestures and thereby becoming independent of mouse or keyboard manipulations to some extent.

5.3. Automating Tasks

Most if not all PC users are dealing with performing repetitive and tedious tasks in their day to day PC interactions such as opening a specific document to work on or browsing their e-mail account. There may often be long sequential commands that are required to be executed in order to accomplish a desired operation and if a user accidentally forgets one of these steps, it may slow the whole process down. By defining these sequences of actions to PersoSys and executing through the mapped gestures, the system both assists the user in not forgetting any steps and expedites the tasks execution leading the users to have a more appealing interaction and minimize their time and energy consumption.

5.4. A Gestural controller for Ordinary PCs

Gestural interface applications have become prevalent and popular especially because of their usage in many devices such as touch enabled computers and smart phones. Recently, the creation of some popular and interactive applications were accompanied with gesture interfaces. These new applications are mapped with special gestures performed in two or three dimensional spaces. Map navigation is, for example, one of the applications that people use to operate and interact with by applying gestures to the touchscreen in order to navigate or zoom the map. When these applications are transferred to other devices without touch sensors, such as the traditional PCs, system designers create graphical widgets to manipulate them. Users however still like to use the same gestures for controlling the applications. In this case, with the use of PersoSys, users may assign their desired gestures to these types of tasks. In other words, it allows the users to transfer the touchscreen gestures to their ordinary PCs and Laptops and personalize the tasks according to their favorite gestures.

CHAPTER 6

Methodology

In this section, we describe the user study we conducted to evaluate PersoSys performance. In this study, we evaluated two primary goals of PersoSys. Those included developing a system that allows users to personalize the behavior of a computer according to their interest and utilizing the systems without significant knowledge about it. We conducted an observational study to understand how adequately the system met these goals when asked to design and conceptualize gestures for sets of tasks. We also measured the performance of the gesture recognizer to determine its accuracy. We asked our participants to consider two sets of tasks and design two gestures for each task, and then define both the gestures and tasks by following the instructions that the GUI of PersoSys provides. Later they executed the mapped tasks by performing the related hand motion gestures.

6.1. Participants

Five participants (2 male, 3 female) were volunteered for this study. All participants were volunteers and were compensated for their time. They ranged in age from 22 to 29. Average was 25.8 years of age. Three of the participants have majors outside of the computer science field. One participant was left-handed, and the others were right-handed. Except for one participant that had some experience using Microsoft Kinect, none of the participants were experienced gestural interface users.

6.2. Setup

The Intel Perceptual SDK camera was placed on top of a computer monitor much like a webcam. The operating system of the computer was Windows 7. The participants were then seated in front of the desktop hosting PersoSys software. They started by defining a new configuration to the systems including creating two pairs of gestures and tasks.

6.3. Procedure

In order to map gestures to tasks, PersoSys asked users to first design the gesture and then execute the tasks by using keyboard and mouse instructions. After completing all the tasks, the subject was directed to a questionnaire to rate subjective experiences on a 10-point Likert scale (Figure 6.1). The questions are listed below:

1. Rate your experience on recording a gesture.
2. Rate your experience on automating a set of tasks.
3. How do you feel about being able to create your own gestures for tasks?
4. How often would you use PersoSys if it was available on your computer system?
5. How would you rate the ability of the PersoSys to recognize your gesture?
6. How would you rate the system overall?

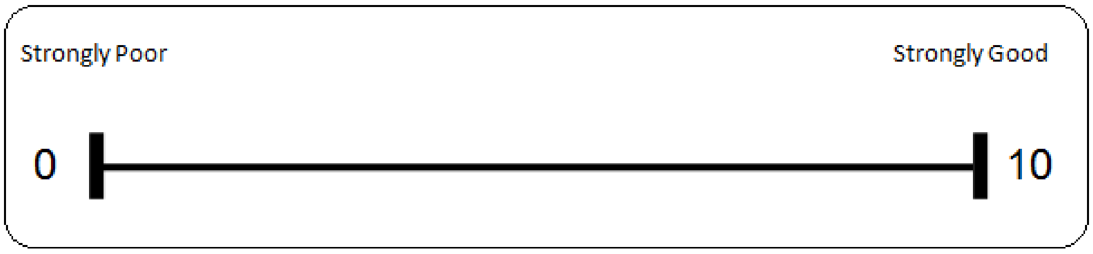


Figure 6.1. 10-points Likert scale

6.4. Results

The responses to the questionnaire for subjective rating are summarized in Figure 6.2. Overall, the results show that participants like the idea of designing their own gestures for tasks and that PersoSys guides them through this process appropriately. They are satisfied with the experience of gesture recording coupled with automating tasks, thus they rated the system positively.

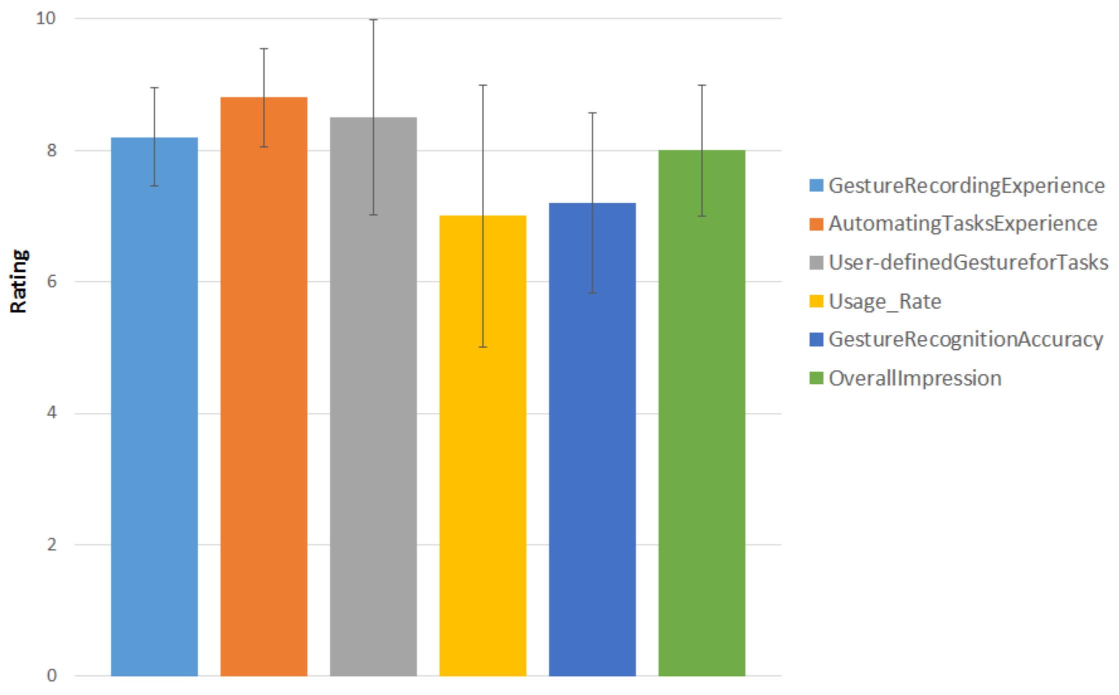


FIGURE 6.2. Mean and standard error of the subjective ratings.

The lowest rate was regarding to the usage. Most of the users specified that they are unable to find a lot of tasks that can be automated using PersoSys. One of the participants commented that;

"Currently, I cannot imagine any long tasks that require clicking so much."(P2)

However, they expressed that if they have to perform the long tasks, automating them would improve the speed and make it so simple and easy.

During the experiment, we observed that one of the participants had a problem setting up their hand in front of the camera. She misunderstood that the system's interface that displays the depth videos is responsible for capturing the hand's points. She thought that she must perform the gesture directly in front of the monitor instead of Intel camera sensors so that the interface would perceive her gestures which consequently led her to locate her hand out of the Intel camera view.

When participants designed their own gestures, it was seen that they prefer short gestures that could be performed in short length of time such as big hand (high five), victory, or waving hand gestures. Further illustrating this, one of the users commented that the two second window used for recording a gesture is a long time;

"I can perform any gesture in less than two seconds."(P4)

In this same session, the participant who has experience using gesture devices asked about the points appearing on the depth video; and they were wondering what these points stand for. When we asked them about the reason they answered:

"I like to know how the sensor captures my gesture so that I can design a perfect gesture that the system can recognize effectively."(P4)

This user commented that they care about the performance of the gesture detection as well as designing their favorite gesture. As such they would not want to design a gesture that is over the ability of the gesture recognizer. This helped us realize that providing a depth video including the detected points of a hand for users during recording session would be helpful for them to design more effective gestures.

CHAPTER 7

Discussion

In this section, we discuss the challenges and potential of utilizing PersoSys in computer interactions based on the results of our study, as well as implications for future software refinement and extensions.

7.1. Understanding End-User Gesture and Task Creation Preferences

Participants were able to successfully create gesture and task maps. The system design, particularly the interface design, was simple and informative enough to guide the users through creating a gesture definition for a task and testing the system. Overall, participants were satisfied with the fundamental concept of the system which allows them to define hand gestures and map them without many limitations.

The previously mentioned incident that happened to one participant, and their confusion about where to locate the hands, prompted a realization that although the interface is largely successful in providing clear detailed instructions to users, there may be some corner cases where it may be helpful to provide more informative instructions to them. It would probably be helpful to users to provide them with a training session video showing the correct way of designing gestures and tasks in terms of appropriate hand position and examples of correctly achieving each step. This video would be displayed for users when they intend to design new gestures and tasks for the system.

The participants' preference for designing short gestures is a reasonable evidence in favor of reducing the length of recording time from two seconds, which in turn allows reducing the recognition time and increasing the performance of the system in terms of time, accuracy, and memory usage.

In the task automation domain, participants had mainly two issues. First, participants liked to have automation for tasks that are highly dependent on a previous set up. For example, minimizing all open windows. The completion of this task is dependent on the window's location on the desktop. Since the tasks recording process saves the motion of the mouse and the activities of the keyboard, it will not execute the minimizing window if the window is located in a different place than the position it was during the recording of the task. This demand seems incredibly popular for users. In Sikuli [29], the users can minimize the windows by capturing a screenshot of the minimize icon. Therefore, when the search engine finds the minimize icon on the screen, it will apply the left-click action on it. However, participant's demands were more specific since they wanted to minimize particular windows like the browser, not all of the windows appearing on the desktop. It seems that this sorts of tasks are a challenge for PersoSys. On the other hand, particpants expressed that PersoSys can play a significant role in automating the tasks which have a static setup. An example could be editing documents on word processing programs such as Microsoft Word. The users can conveniently change the page layout or manipulate the line spacing options, the fonts, colors, and etc. Making PersoSys intelligent enough to execute tasks independently of the state they were originally recorded in is a desired feature that we will focus on in the future.

The second issue in task automation domain that participants expressed was designing gestures for the special tasks that seemed to map naturally to gestures such as application movement.

"I'd like to drag and move the open windows with my hand like dragging objects with my hand in reality."(P3)

Currently, PersoSys is unable to achieve these sorts of actions mainly because the system records the tasks in terms of mouse and keyboards actions and it does not simulate the gestures that are performed in real life. However, it motivates us to think about adding more capability to PersoSys, including controlling and executing tasks that are more connected to gesture interaction and developing them in the future.

Although participants had these two challenges in PersoSys usage, they expressed that this system would be a great help for them to automate long and tedious tasks that take time to do manually by mouse and keyboard. The feedback also indicated that PersoSys has massive potential if it can be expanded to more dynamic tasks.

7.2. Limitations

In PersoSys development, we employed the 2013 Intel SDK sensor which captures 11 points of hand and joint tracking. This limited number of points causes the gesture recognition method to make some mistakes with detecting hand gestures that are similar to each other. An example is when a participant designs a static big hand (high five) gesture as well as a big hand gesture with moving fingers. In these gestures, the only difference is moving the fingers which is hard for system to distinguish. If the Intel camera could capture more points on a finger, then the gesture recognition method of system would overcome this problem and detect the similar gestures more precisely. Fortunately, in 2014 the Beta SDK version of the Intel camera captures 22 points of hand and joint tracking instead of 11 points [1]. Since PersoSys software was implemented before releasing the 2014 Intel SDK camera, we were not able to apply these new features in our system. Future work needs to be done to employ advanced hardware in PersoSys.

CHAPTER 8

Conclusion and Future Work

We have presented PersoSys, which allows users to define a personalized set of hand gestures that best fit particular sets of computing tasks and create a mapping from hand gestures to system behavior in a way that offers a more satisfactory user experience than with presently predefined gestural interfaces. In addition to customizing the gesture interactions, our system enables users to assign flexible tasks including a long sequence of tasks to a gesture. The system recognizes the performed user-defined gestures and triggers an event to execute the assigned tasks automatically which may otherwise be tedious and time-consuming if the user were forced to perform the tasks manually by using traditional peripheral devices. Our observational experiment also confirmed that users are pleased by controlling computers and running tasks automatically by hand gesture input commands.

8.1. Future Work

During the experiment, we learnt that many users have challenges with a variety of task automations, especially the manipulation of GUI elements whose executions are highly dependent on their location on screen during the training session. A possible solution would be to implement a function that would detect the *type* of event that is triggered as well as the application that the events performed on in addition to merely recording the mouse and keyboards actions.

In this work, we have considered PCs and Laptops as the main application scenario of PersoSys. However, we believe that our approach can also be applied to other smart, personal devices that can be operated by gesture interactions and adapted personalization and customization of system behavior. Examples include televisions, game controls, smart mobile phones, tablets, and assistive devices for handicapped people. In order to apply the PersoSys strategy to other devices, they need merely be capable of communication with one of the various gesture recognition sensors or contain them in their platforms. For the devices that has different sensor than Intel SDK camera, PersoSys can be applied to them by minor changes in the gestures recognition component.

Bibliography

[1] Intel realsense sdk for windows beta product release questions. `http://software.intel.com/sites/default/files/IntelRealSenseSDKforWindowsBetaFAQ.pdf`.

[2] Bigdelou, A., L. Schwarz, and N. Navab (2012). An adaptive solution for intra-operative gesture-based human-machine interaction. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, New York, NY, USA, pp. 75–84. ACM.

[3] Codd-Downey, R. and W. Stuerzlinger (2014). Leaplook: A free-hand gestural travel technique using the leap motion finger tracker. In *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*, SUI '14, New York, NY, USA, pp. 153–153. ACM.

[4] Cohé, A. and M. Hachet (2012). Understanding user gestures for manipulating 3d objects from touchscreen inputs. In *Proceedings of Graphics Interface 2012*, GI '12, Toronto, Ont., Canada, Canada, pp. 157–164. Canadian Information Processing Society.

[5] de Barros, A. C., J. a. Cevada, A. Bayés, S. Alcaine, and B. Mestre (2013). User-centred design of a mobile self-management solution for parkinson's disease. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, New York, NY, USA, pp. 23:1–23:10. ACM.

[6] Hoc, J. and A. Nguyen-Xuan (1990). *Psychology of Programming.* Reading, Massachusetts: European Association of Cognitive Ergonomics and Academic Press.

[7] Kane, S. K., J. O. Wobbrock, and R. E. Ladner (2011). Usable gestures for blind people: Understanding preference and performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, New York, NY, USA, pp. 413–422. ACM.

[8] Kurdyukova, E., M. Redlin, and E. André (2012). Studying user-defined ipad gestures for interaction in multi-display environment. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, New York, NY, USA, pp. 93–96. ACM.

[9] Klsch, M., R. Bane, T. Hllerer, and M. Turk. Multimodal interaction with a wearable augmented reality system. *Computer Graphics and Applications, IEEE*.

[10] Liang, H.-N., C. Williams, M. Semegen, W. Stuerzlinger, and P. Irani (2012). User-defined surface+motion gestures for 3d manipulation of objects at a distance through a mobile device. In *Proceedings of the 10th Asia Pacific Conference on Computer Human Interaction*, APCHI '12, New York, NY, USA, pp. 299–308. ACM.

[11] Liu, J., L. Zhong, J. Wickramasuriya, and V. Vasudevan (2009, December). uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive Mob. Comput. 5*(6), 657–675.

[12] Markussen, A., M. R. Jakobsen, and K. Hornbæk (2014). Vulture: A mid-air word-gesture keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, pp. 1073–1082. ACM.

[13] Mauney, D., J. Howarth, A. Wirtanen, and M. Capra (2010). Cultural similarities and differences in user-defined gestures for touchscreen user interfaces. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, New York, NY, USA, pp. 4015–4020. ACM.

[14] Merrill, D. J. and J. A. Paradiso (2005). Personalization, expressivity, and learnability of an implicit mapping strategy for physical interfaces. *Extended Abstracts of CHI 2005*.

[15] Morris, M. R., J. O. Wobbrock, and A. D. Wilson (2010). Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010*, GI '10, Toronto, Ont., Canada, Canada, pp. 261–268. Canadian Information Processing Society.

[16] Nancel, M., J. Wagner, E. Pietriga, O. Chapuis, and W. Mackay (2011). Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, New York, NY, USA, pp. 177–186. ACM.

[17] Oh, U. and L. Findlater (2013). The challenges and potential of end-user gesture customization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, New York, NY, USA, pp. 1129–1138. ACM.

[18] Pavlovic, V. I., R. Sharma, and T. S. Huang (1997, July). Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell. 19*(7), 677–695.

[19] Pisharady, P. K. and M. Saerbeck. Robust gesture detection and recognition using dynamic time warping and multi-class probability estimates. *Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), 2013 IEEE Symposium.*

[20] Piumsomboon, T., A. Clark, M. Billinghurst, and A. Cockburn (2013). User-defined gestures for augmented reality. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, New York, NY, USA, pp. 955–960. ACM.

[21] Ruiz, J., Y. Li, and E. Lank (2011). User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, New York, NY, USA, pp. 197–206. ACM.

[22] Scaffidi, C., J. Brandt, M. Burnett, A. Dove, and B. Myers (2012). Sig: End-user programming. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, New York, NY, USA, pp. 1193–1996. ACM.

[23] Vatavu, R.-D. (2012). User-defined gestures for free-hand tv control. In *Proceedings of the 10th European Conference on Interactive Tv and Video*, EuroiTV '12, New York, NY, USA, pp. 45–48. ACM.

[24] Vatavu, R.-D. and I.-A. Zaiti (2014). Leap gestures for tv: Insights from an elicitation study. In *Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '14, New York, NY, USA, pp. 131–138. ACM.

[25] Walter, R., G. Bailly, N. Valkanova, and J. Müller (2014). Cuenesics: Using mid-air gestures to select items on interactive public displays. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices &#38; Services*, MobileHCI '14, New York, NY, USA, pp. 299–308. ACM.

[26] Wobbrock, J. O., M. R. Morris, and A. D. Wilson (2009). User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, New York, NY, USA, pp. 1083–1092. ACM.

[27] Xiao, R., C. Harrison, and S. E. Hudson (2013). Worldkit: Rapid and easy creation of ad-hoc interactive applications on everyday surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, New York, NY, USA, pp. 879–888. ACM.

[28] Yan, X. and N. Aimaiti. Gesture-based interaction and implication for the future. master thesis, Department of Computer Science, Umea University, Umea, Sweden.

[29] Yeh, T., T.-H. Chang, and R. C. Miller (2009). Sikuli: Using gui screenshots for search and automation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, New York, NY, USA, pp. 183–192. ACM.

[30] Zettlemoyer, L. S. and R. S. Amant (1999). A visual medium for programmatic control of interactive applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, Pittsburgh, Pennsylvania, United States, pp. 199–206. ACM Press.