A Multi-touch Solution to Build Personalized Interfaces for the Control of Remote Applications

Gianluca Paravati[1], Mattia Donna Bianco[2], Andrea Sanna[1], and Fabrizio Lamberti[1]

[1] Politecnico di Torino, Dipartimento di Automatica e Informatica,
C.so Duca degli Abruzzi 24, I-10129, Torino, Italy
[2] CEDEO.net, Via Borgionera 103, I-10040, Villar Dora (TO), Italy
{gianluca.paravati,andrea.sanna,fabrizio.lamberti}@polito.it,
mattia@cedeo.net

Abstract. This paper presents a framework for controlling remote applications by means of personalized multi-touch interfaces. The designed framework allows end-users to fully personalize the mapping between gestures and input commands. A two-tier architecture has been developed. A formal description of the original interface is automatically generated at the server side to identify a set of available actions for controlling existing applications. The client is in charge of loading the description of the target application, allowing the user to shape the preferred mapping between gestures and actions. Finally, the server converts the identified actions into one or more commands understandable by the original computer interface. The implementation of the system for this work specifically relies on handheld multi-touch devices. Test results are encouraging, both from an objective and a subjective point of view; indeed, the designed framework resulted to outperform a traditional GUI both in terms of number of actions to perform a task and average completion time.

Keywords: Multi-Touch, personalized interfaces, human-machine interface, remote control.

1 Introduction

Human-machine interaction based on touch devices is quite common today. The evolution of input device technologies such as reflection-based or pressure-sensitive touch surfaces led to identification of the natural user interface (NUI) as the clear evolution of the human-machine interaction, following the shift from command-line interfaces (CLI) to graphical user interfaces (GUI).

The main goal of human-machine interaction is to improve the way users and computers communicate, by means of effective user interfaces. The design of user interfaces requires a careful mapping of complex user actions in order to make computers more intuitive, usable and receptive to the user's needs: in other words, more user-friendly.

Gestures, and in particular hand gestures, ever played a crucial role in human communication, as they constitute a direct expression of mental concepts [1]. The naturalness and variety of hand gestures, compared with traditional interaction paradigms, can offer unique opportunities also for new and attracting forms of human-machine interaction [2]. Thus, new gesture-based paradigms are progressively introduced in various interaction scenarios (encompassing, for instance, navigation of virtual worlds, browsing of multimedia contents, management of immersive applications, etc. [3][4][21]), and the design of gesture-based systems will play an important role in the future trends of the human-computer interaction.

Indeed, we use gestures to express ourselves most of the times. Thus, GUI-based applications could be converted to take as input intuitive gestures. By means of a mapping between the original interface and the gesture-based one, applications could gain a higher degree of user friendliness, achieve better performance, and become easier to interact with. This new paradigm started to be introduced also for the control of consumer electronic devices. For instance, the CRISTAL project (Control of Remotely Interfaced Systems using Touch-based Actions in Living spaces) [5] enables people to control a wide variety of digital devices (TVs, music players, digital picture frames, speakers, light sources, etc.) from an interactive tabletop system that provides users with a gesture-based interface. User controls the devices through a virtually augmented representation of the surrounding environment.

Despite the evolution of the mobile world, most of the existing applications designed for the desktop world cannot run on mobile devices due to different hardware and graphics capabilities. One solution for allowing mobile user to access desktop-like applications involves the use of remote-computing techniques, where a remote server executes a specific application sending to the mobile device only the relevant graphics representation through a sequence of still images or a video stream [6][7]. This solution allows users to remotely control virtually any kind of application, including those that are still out of reach for handheld devices (as, for instance, the navigation in a 3D world composed by millions of polygons). Recently, a software independent approach extending the basic remote control paradigm to maintain a separate work area and user interface has been presented [8].

The aim of the proposed work is to describe a user-centric methodology to support the generalization of the mapping between existing GUIs and up-to-date NUIs, thus matching the above needs. To this purpose, the approach described in [8] is extended to build a solution supporting the creation of personalized and customizable human computer interaction interfaces by means of a generic gesture-based framework based on the multi-touch technology. The main advantage of the designed solution is that it allows for the development of user interfaces that most effectively and intuitively leverage one of the more relevant senses into the most optimal user friendliness (for instance, with projected-capacitive touch screen, users can give more complicated inputs like those used for sizing photos, adjusting web pages, etc.).

Basically, the framework is structured into a two-tier architecture. On one side, a server component is in charge of managing any kind of desktop application and to automatically build the description of its interface. This description is used to identify the functionalities of the target application and to create a personalized mapping between functionalities and user gestures that is then used for application control. Because of the availability of the 802.11g connectivity and of a mature set of APIs, in our test-bed the client side was implemented on the Apple's iPod Touch, one of the most popular multi-touch devices currently available on the market. However, the proposed approach is generic and can be applied to any kind of device able to capture user gestures.

This paper has been organized as follows. Section 2 reviews main works related to multi-touch interfaces. The proposed framework is described in Section 3. Section 4 presents a case study and the results of objective and subjective evaluations given by a set of users. Finally, conclusions and future works are discussed in Section 5.

2 Background

Recently, multi-touch technology has gained increasing attention both in the research community and the commercial world. On the market, various devices able to capture gestures already exist, ranging from tabletop displays to handheld devices. An in depth discussion on the application of multi-touch techniques for providing full control (6-DOFs) of a 3D rigid body using tabletop displays is reported in [9]. A different approach exploiting tabletop surfaces for designing a more realistic and sophisticated form of interaction is presented in [10].

The introduction of devices such as the Apple's iPhone and iPod Touch led researchers to focus their attention towards multi-touch user interfaces tailored to mobile environments. For instance, a framework to dispatch multi-touch events generated on a mobile device to a tabletop system is presented in [11]. Collaboration among users equipped with handheld multi-touch devices and tabletop frameworks is investigated in [12]. An interesting approach for manipulating 3D objects on multi-touch mobile devices has been recently presented in [13]. In this work, two iPod Touch units are attached back to back, and connected through a Wi-Fi connection; in this way, the freedom of manipulation is extended to a (pseudo) 3D scenario obtained by the sandwiched fixed volume architecture [14]. In [13] and [21], preliminary evaluations of the proposed interfaces are also carried out by collecting end-user feedbacks.

Gesture input on multi-touch handheld devices has not been used only for the control of desktop applications. As a matter of example, applications exploiting multi-touch technology to control IR devices are already available on the market. For instance, the RedEye system allows users to directly control devices such as TV, stereo, cable box, DVD player, and many other units that receive standard infrared signals by means of their iPhone or iPod Touch devices[15]. Similarly, in [16] iPhone and iPod Touch platforms are used for remote sensor control and data collection.

An interesting approach to adapt gesture input to the controlled application has been taken by the developers of SparshUI [17]. SparshUI is a platform-independent framework for developing multi-touch enabled applications, composed by a gesture server (in charge for handling the gesture processing), a gesture adapter (which is different for every controlled application), and an input device driver (that is needed to communicate with the gesture server). Drivers for several types of hardware devices have already been developed. Similarly, a software architecture supporting multi-touch capability on existing desktop systems, where both multi-touch and multiple single pointer inputs can be used simultaneously to manipulate existing application windows is proposed in [18]. The authors presented a proof-of-concept implementation of their architecture on the Linux system, demonstrating the possibility of using touch displays in a collaborative work environment.

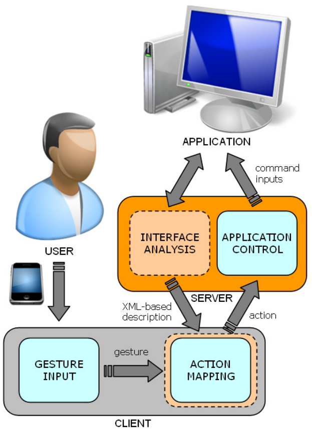

Fig. 1. Conceptual architecture of the designed multi-touch remote control system

The work presented in this paper follows a concept similar to SparshUI. However, it is aimed at enabling users to create their own multi-touch applications, whereas our intention is to adapt already existing applications to the multi-touch technology; for this reason, we need to create a description of the original interface and we lean on the results obtained in [8], where a software-independent framework, able to exploit image processing techniques to effectively decompose an original application into its main GUI elements, create a description of the original interface, and reload a personalized GUI on different devices, is presented.

3 The Proposed Framework

The proposed framework is structured into the two-tier architecture that is depicted in Figure 1. The user directly interacts with the client component, which is demanded to deal with all the aspects concerning gesture identification, interpretation, and personalization. On the other hand, the server interacts with the controlled application providing it with the translated gestures. The proposed solution requires a setup phase (indicated in Figure 1 by the dashed boxes), followed by the actual gesture mapping chain, composed by three steps, namely “Gesture Input”, “Action Mapping” and “Application Control”.

The setup phase is meant to create a formal description of the target application interface and it is composed by an off-line step (i.e. “Interface Analysis”) and an on-line step (i.e. a part of the “Action Mapping” step). The off-line step is introduced for allowing the client application to know in advance which actions are associated to the target application. During the second step of the setup phase, the user will be able to personalize the mapping between gestures and available actions.

The interface of an existing application can be automatically analyzed through reverse engineering approaches in order to build a description of its elements. In this work, an image-based approach proposed in [9] has been used as the basis to design the “Interface Analysis” phase, which provides a method to create an XML-based description of the elements belonging to the GUI, i.e. the concrete aspects. Since other kinds of input methods can be possibly used to control the application, such as mouse or keyboard events, the method in [9] has been extended by introducing a language able to overcome the above issues. UsiXML [16], a XML-compliant User Interface Description Language, was used for this purpose: it is aimed at describing user interfaces according to four levels of abstraction: task model, abstract user interface, concrete user interface and final user interface. The two models describing tasks (i.e. actions) and the concrete aspects (buttons, keys, etc.) are of particular interest for the purpose of this work. The first model is described using UsiXML. The tasks can be composed by more sub-tasks; moreover, UsiXML allows to specify relationships among the tasks. The process of creating the complete description ends with the definition of a mapping between the two models in order to link them.

During the on-line step of the setup phase, the client device receives from the server a list of available applications and the corresponding XML-based descriptions. Once the target application has been selected, its XML-based description is loaded on the client device and the user can select a personalized gesture for each action. This step of the setup phase is aimed at creating a mapping between the multi-touch gestures (e.g. single, double, or multiple tap, pinch-in, pinch-out, joined or separate fingers, clockwise or counterclockwise rotations, etc.) and the described actions, which is stored into a conversion table. Moreover, during the setup phase the personalization of the multi-touch remote control is improved by allowing the user to select the preferred sensibility settings for each gesture.

The gesture mapping chain is split into two branches: “Gesture Input” and “Action Mapping” take place on the client device, whereas the “Application Control” logic is located on the server side. “Gesture Input” is oriented towards the human interface, taking as input the gestures drawn by the user on the multi-touch screen and delivering them to the next steps. During the intermediate step each recognized gesture is then translated into a meaningful action in the application context, based on the conversion table defined by the user during the setup phase, thus personalizing the new interface. On the other hand, the “Application Control” is part of the computer interface; each action received from the previous conversion block can be mapped into one or more input commands, which are then delivered to the target application. The original input can be either discrete (e.g., key presses) or continuous (e.g., mouse position, although digitized into a discrete quantity, is fast enough to be considered as continuous). To complete the description of the framework, a swim lane diagram showing, step-by-step, tasks involved in a client-server connection is presented in Figure 2.

At an early stage, a custom communication protocol has been defined for the management of all the events generated in the considered architecture, i.e. for the handling of the gesture association stage, the selection of the target application, the delivery of actions from the multi-touch device to the server, etc. However, it is worth observing that hardware devices and touch libraries are progressively adopting TUIO, an open standard recently introduced for structuring the description of touch event based communications [20] developed to provide hardware-independence. Therefore, it has been already planned the experimentation of the TUIO framework in the designed architecture.

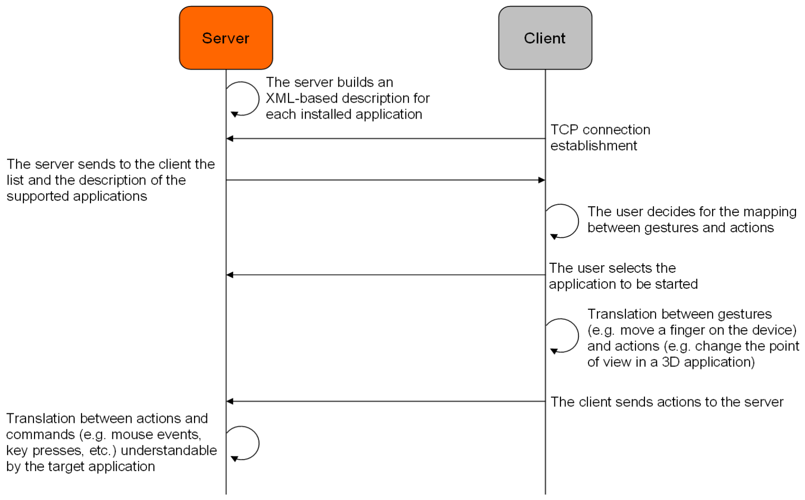
Server
Client
The server builds an XML-based description for each installed application
TCP connection establishment
The server sends to the client the list and the description of the supported applications
The user decides for the mapping between gestures and actions
The user selects the application to be started
Translation between gestures (e.g. move a finger on the device) and actions (e.g. change the point of view in a 3D application)
The client sends actions to the server
Translation between actions and commands (e.g. mouse events, key presses, etc.) understandable by the target application

Fig. 2. Swim lane diagram of tasks involved in a client-server connection

4 A Case Study

The proposed framework has been implemented as a client-server application. The server application runs on a desktop PC with Microsoft Windows XP; with reference to Figure 1, the “Interface Analysis” and “Application Control” modules have been implemented in C++ language to be platform independent. The minimum requirements of the computer are determined by the target application to be controlled, since the server application has a very small footprint. The client application runs on an Apple iPodTouch device; it has been implemented in Objective-C language and Cocoa, the Apple’s native object-oriented application development framework. The communication between the client and the server application occurs via a TCP connection and the handheld multi-touch device is connected to the desktop PC through a 802.11g wireless access point.

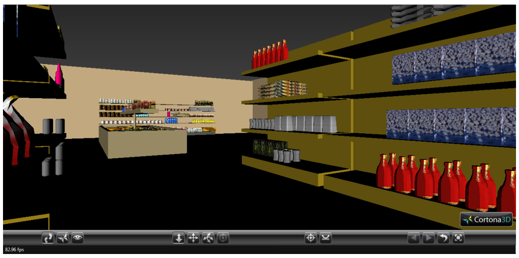

Fig. 3. Snapshot of the supermarket model used for testing the multi-touch interface

The evaluation process of the proposed framework was aimed at assessing the added value of the multi-touch interface for an existing generic application. For this purpose, the designed solution was tested and evaluated both from an objective and a subjective point of view by a group of forty-three end-users. A 3D viewer has been chosen as the target application, due to its intrinsic interactivity constraints; indeed, in this kind of applications each action require a closed-loop control by the user. The test consisted in navigating a 3D supermarket environment, searching for a specified shopping list including five objects (http://conferences.computer.org/3dui/3dui2010/). The test was considered completed when all the objects had been collected. The objects were placed all around the supermarket, in order to force the users to walk through the scene for at least two minutes. Moreover, the objects were placed on shelves of different heights; thus, it was necessary to look up and down to grab them. The 3D viewer application selected as a case study was Cortona3D (http://www.cortona3d.com/). A screenshot of the application during a test sequence is shown in Figure 3. User population was mostly composed by students of a Computer Graphics course, i.e. it represented a quite homogeneous group: users have similar background, interests and knowledge. On the other hand, it is worth to remark that the selected population could be not completely representative; in fact, students of the above course are already used to work with 3D computer graphics applications.

Only ten out of forty-three users owned a touch device. As their performances did not stand out, in the following they will not be discussed as a different group, but they will be kept together with the other.

Since the main interest during the case study was to evaluate the impact of the new user interface, the correlation due to the influence of the personalization of user gestures was minimized by choosing in advance default values both for sensibility and gesture mapping settings. Each user was individually trained on the execution of the test; then, each user was requested to complete the test both with the multi-touch device and the mouse. The tests were submitted with a random order, in order to limit the impact of the knowledge acquired during the first attempt on the overall result.

The overall usability and effectiveness of the experimented interface can be inferred from Table 1. The objective results were gathered focusing on the analysis of the average completion time and the average number of interactions needed to complete the task. In particular, each gesture was considered as a single interaction as well as each press-release of a button or key stroke, thus allowing to achieve a fair analysis of the two interfaces. The gesture-based framework resulted to outperform the traditional interface, both in terms of average completion time (13% improvement) and number of actions (18% saving).

The interest was not only on the objective data, but also on the subjective information (i.e., the impression that students had using the proposed interface). At the end of any test session, each user was asked to compile a brief questionnaire about the personal evaluation of the interfaces; in particular, the users were allowed to give a score ranging from one to four, with one indicating a forced/unnatural interface, and four meaning a very intuitive one. As shown in Table 1, the gesture-based solution appears to be significantly more user friendly than the original interface.

Table 1 also shows a statistical analysis on the experimental data that has been carried out to the purpose of disclosing the confidence level of the results obtained with the gesture based (GB) and the traditional (TR) interfaces. As variances are unknown and small samples are taken, the t-statistic was used to test the differences between the interfaces.

A paired t-test was performed by testing the hypothesis that the mean of differences between each pair of observation μ_t was μ_{GB}-$\mu_{TR}=0$, both in terms of completion time, total number of interactions and subjective scores. As a rule of thumb, the risk level for computing the reference t-value for comparison was set to $\alpha=5\%$, finding a reference t-value equal to 1.99. According to the statistical analysis, as shown in Table 1, the t-statistic values computed for the comparison of the average number of actions and the subjective results were larger than the reference t-value; therefore, the null hypothesis could be rejected. Moreover, since the confidence level was larger than 99%, the proposed framework definitely showed to outperform the traditional interface. On the other hand, the t-statistic value computed for the time comparison resulted to be lower than the reference t-value. As a matter of fact, in the latter case the null hypothesis (with $\alpha=5\%$) could not be rejected. Although the decrease in user actions and the increase in the degree of user satisfaction could be specifically related to the adoption of the gesture-based framework, the saving in time might be not fully related to the same fact (a confidence level equal to 70% was reached). This could be explained by considering the relation between the type of tasks that users needed to perform and the size of the input device: as already outlined in [21], actions that need a high level of precision can not be accurately controlled by multi-touch mobile device.

Table 1. Objective and subjective results indicating the performance using the traditional interface and the gesture-based framework. All the results are expressed as average values.

Interface	*Objective Results*		*Subjective Results*
	Time	*Actions*	*User evaluation*
Traditional	252 s	138	2.65
Gesture-based	218 s	113	3.23
Statistics:			
Gesture-based improvement	13%	18%	18%
t-statistic value	1.05	3.69	3.59
confidence level	70%	99%	99%

5 Conclusions and Future Works

A customizable and portable human computer interaction interface has been presented in this paper. The framework exploits a gesture-based paradigm relying on the multi-touch technology and it is aimed at controlling existing applications by translating the original command inputs into gestures that can be customized by the user. The main reason behind the measured improvement is that by using gestures an immediate and more intuitive access to the action to be performed can be achieved.

Future works will be aimed at investigating the use of gesture-based interfaces to control real-time applications (e.g. to supervise a robot). Moreover, further experiments involving larger multi-touch input devices are planned, with the aim of checking if it is possible to reduce the limitations identified during the execution of high precision tasks.

Acknowledgment. This article is part of a work developed within the frame of the project "Piattaforma Tecnologica Innovativa per l'Internet of Things" co-funded by the Regione Piemonte.

References

1. Pavlovi'c, V.I., Sharma, R., Huang, T.S.: Visual interpretation of hand gestures for human-computer interaction: a review. IEEE TPAMI 19, 677–695 (1997)
2. Pavlovi'c, V.I., Sharma, R., Huang, T.S.: Gestural interface to a visual computing environment for molecular biologists. In: Proc. of the 2nd Intern. Conf. on Automatic Face and Gesture Recognition, pp. 52–73. IEEE Computer Society, Los Alamitos (1996)
3. Selker, T.: Touching the future. Commun. ACM 51, 14–16 (2008)
4. Wright, A.: Making sense of sensors. Commun. ACM 52, 14–15 (2009)
5. Seifried, T., Rendl, C., Perteneder, F., Leitner, J., Haller, M., Sakamoto, D., Kato, J., Inami, M., Scott, S.D.: CRISTAL, control of remotely interfaced systems using touch-based actions in living spaces. In: SIGGRAPH 2009 Emerging Technologies, N.Y. (2009)
6. Gong, J., Tarasewich, P.: Guidelines for Handheld Mobile Device Interface Design. In: Proc. Decision Sciences Inst., Decision Sciences Inst., pp. 3751–3756 (2004)
7. Florins, M., Vanderdonckt, J.: Graceful Degradation of User Interfaces as a Design Method for Multiplatform Systems. In: Proc. 9th ACM Int'l Conf. IUI 2004, pp. 140–147 (2004)
8. Lamberti, F., Sanna, A.: Extensible GUIs for Remote Application Control on Mobile Devices. IEEE Computer Graphics and Applications 28(4), 50–57 (2008)
9. Hancock, M., Carpendale, S., Cockburn, A.: Shallow-depth 3D interaction: design and evaluation of one, two and threetouch techniques. In: CHI 2007: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1147–1156. ACM, N.Y (2007)
10. Wilson, A.D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., Kirk, D.: Bringing physics to the surface. In: UIST 2008: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, pp. 67–76. ACM, New York (2008)
11. Hafeneger, S., Weiss, M., Herkenrath, G., Borchers, J.: Pockettable: Mobile devices as multi-touch controllers for tabletop application development. Extended Abstracts of Tabletop 2008 (2008)

12. Nestler, S., Echtler, F., Dollinger, A., Klinker, G.: Collaborative problem solving on mobile hand-held devices and stationary multi-touch interfaces. In: PPD 2008: Workshop on Designing Multitouch Interaction Techniques for Coupled Public and Private Displays (2008)
13. Shen, E.L., Tsai, S.S., Chu, H.H., Hsu, J., Chen, C.W.: Double-side multi-touch input for mobile devices. In: CHI 2009: Proceedings of the SIGCHI. ACM, New York (2009)
14. Wigdor, D., Leigh, D., Forlines, C., Shipman, S., Barnwell, J., Balakrishnan, R., Shen, C.: Under the table interaction. In: UIST 2006: Proc. of the 19th Annual ACM Symposium on User Interface Software and Technology, pp. 259–268. ACM, New York (2006)
15. RedEye, `http://thinkflood.com/products/redeye/what-is-redeye/`
16. Geltz, B.R., Berlier, J.A., McCollum, J.M.: Using the iPhone and iPod Touch for remote sensor control and data acquisition. In: IEEE Proc. of the SoutheastCon, pp. 9–12 (2010)
17. Sparsh-UI, `http://code.google.com/p/sparsh-ui/`
18. Cheng, K., Itzstein, B., Sztajer, P., Rittenbruch, P.: A unified multi-touch & multi-pointer software architecture for supporting collocated work on the desktop. Technical Report ATP-2247. NICTA, Australian Technology Park, Sydney, Australia (2009)
19. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In: Feige, U., Roth, J. (eds.) DSV-IS 2004 and EHCI 2004. LNCS, vol. 3425, pp. 200–220. Springer, Heidelberg (2005)
20. Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: TUIO: A Protocol for Table-Top Tangible User Interfaces. In: 6th International Gesture Workshop (2005)
21. Fiorella, D., Sanna, A., Lamberti, F.: Multi-touch user interface evaluation for 3D object manipulation on mobile devices. Journal on Multimodal User Interfaces (2009)