ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

TOUCH-FREE USER INTERFACE FOR AUGMENTED REALITY SYSTEMS

ANKUSH RAI*, JAGADEESH KANNAN R

School of Computing Science and Engineering, VIT University, Chennai, Tamil Nadu, India. Email: ankushressci@gmail.com

Received: 13 December 2016, Revised and Accepted: 03 April 2017

ABSTRACT

Augmented reality is the upcoming field of research and is often suffer from the current form of user interface. In this study, we present touch-free user interactive system for augmented reality applications to carry out multi-task operations. We validated the efficacy of the method based on the performance of several users while carrying out the complex task in our sample augmented reality game.

Keywords: Multimedia, Interactive graphical user interface, Automation.

© 2017 The Authors. Published by Innovare Academic Sciences Pvt Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4. 0/) DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19683

INTRODUCTION

In human-to-computer interaction, interacting for information about tangible objects within the computing media is usually accomplished by making indirect visual references to it. On the other hand, the human-to-human interaction with information is accomplished by the visual references which are addressed as the perception. Therefore, in this study, we meant to adjoin the two phenomenon of interaction adding depth to the graphical user interface (GUI) elements. Giving visual commands, finding information or issuing commands involving tangible objects can also be naturally accomplished by making similar visual references as found appropriated by the user. Unlike in system dependent on command line interface (CLI), a GUI has several test operations to successful pass on [1]. Thus, the testing techniques mainly employed for testing CLI programs suffer from scaling problems such as finite state machine when applied in the world of GUI's [2,3]. Therefore, it is required that a different approach is to be used for testing GUI's from what it is employed for CLI technique [4]; which in turn involves usage of a planning system [5,6]. Although employing the technique of capture-playback, functions well in CLI world but often are prone to problems which are quite significant when it is implemented for a GUI system [7]. Accordingly, to eliminate such problems, there were two ways mainly used by the testers; either the testers use data collected from GUI interaction through the underlying windowing system [8], or to build the driver into the GUI such that the commands or events can be dispatched from other programs [9].

However, in the past, there is a field namely gesture control attempts made to achieve this through various computer vision algorithm over various platforms but requires hefty computing even for a simple job of identifying the gesture from real world to the trained gesture configuration from artificial neural networks or other various pattern recognition algorithms which limits its real-time application. In some cases, few device manufacturing companies have developed many-core mobile computing devices where such gesture control features are provided; even though, it is not a cost-effective method cost more battery usage with delayed interactive actions. Thus, there is a need to eliminate such limitations; therefore, we have used the attribute based level adaptive thresholding algorithm for object extraction (ABLATA) [10]; which is implemented in augmented form with the proposed evolutionary algorithm to match up with the user defined motion inputs and classify to learn and imitate the process simultaneously.

MOBILE EVOLUTIONARY ALGORITHM FOR EVASIVE INTERACTIVE GUIS

Algorithm

Input: Two images (a) Instance of window's workspace W (b) instance of user's end U

Output: Action sets AS_i and matrix model of tree of actions M_x

Step 1: Perform ABLATA over U and W; such hat, we get L_U and L_W which are the set of levels from U and W, respectively.

Step 2: Compute the pointing correlation state P as

$$P=\frac{1}{L_N}\sum_{p_t}^{L_W-1}\left[\sum_{p_2}^{L_U-1}S_{p_1,p_2}(t_i,f_1,f_2)\right]\left[\sum_{p_2}^{L_U-1}S'_{p_1,p_2}(t_i,f_1,f_2)\right]$$

where, L_N are the universal set of level for the images, p_i and p_2 are the adjoint pixels intersection with the levels L_W and L_U respectively, $S'_{p1,p2}$ and $S'_{p1,p2}$ are the sets of pixel density constraint layout for the pixel positioning with its patterning saved in levels and between its intersection of adjoint pixels and the super positioned pixel density layout of differing state at the users instance of the image U. Furthermore, t_i is the collection of patterns for the weighted superposed state P_c (Initially its value is set to 0), f_1, f_2 are the two delay frames with a minimal time delays t_i [11-13].

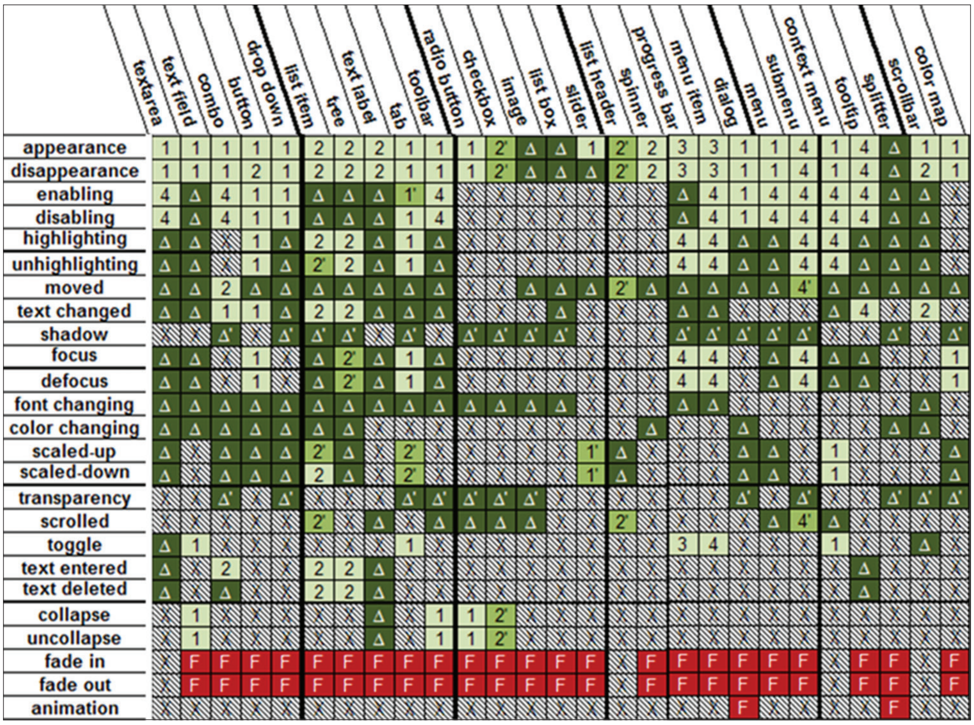
Step 3: For p_{i+1} evaluate

$P \rightarrow P_c$

Step 4: Calculate the tree of action based on continuous feedback loop

$$M_x=\begin{pmatrix} t_1\begin{bmatrix}P_1\\P_4\\P_8\end{bmatrix}=AS_1 \\ t_2\begin{bmatrix}P_3\\P_9\\P_6\end{bmatrix}=AS_2 \\ t_3\begin{bmatrix}P_2\\P_5\\P_7\end{bmatrix}=AS_3 \\ \vdots \\ t_i\begin{bmatrix}P_0\\P_5\\P_c\end{bmatrix}=AS_i \end{pmatrix}$$

Fig. 1a: The performance test of graphical user interface visual behavior under various action sets



	textarea	text field	combo	button	drop down	list item	tree	text label	tab	toolbar	radio button	checkbox	image	list box	slider	list header	spinner	progress bar	menu item	dialog	menu	submenu	context menu	tooltip	splitter	scrollbar	color map
appearance	1	1	1	1	1	2	2	2	1	1	1	2'	Δ	Δ	1	2'	2	3	3	1	1	4	1	4	Δ	1	1
disappearance	1	1	1	2	1	2	2	2	1	1	1	2'	Δ	Δ	Δ	2'	2	3	3	1	1	4	1	4	Δ	2	1
enabling	4	Δ	4	1	1	Δ	Δ	Δ	1'	4	X	X	X	X	X	X	X	Δ	4	1	4	4	4	4	Δ	Δ	X
disabling	4	Δ	4	1	1	Δ	Δ	Δ	1	4	X	X	X	X	X	X	X	Δ	4	1	4	4	4	4	Δ	Δ	X
highlighting	Δ	Δ	X	1	Δ	2	2	Δ	1	Δ	X	X	X	X	X	X	X	4	4	Δ	Δ	4	4	Δ	Δ	Δ	X
unhighlighting	Δ	Δ	X	1	Δ	2'	2	Δ	1	Δ	X	X	X	X	X	X	X	4	4	Δ	Δ	4	4	Δ	Δ	Δ	X
moved	Δ	Δ	2	Δ	Δ	Δ	Δ	Δ	Δ	Δ	X	X	Δ	Δ	Δ	2'	Δ	Δ	Δ	Δ	Δ	4'	Δ	Δ	Δ	Δ	Δ
text changed	Δ	Δ	1	1	Δ	2	2	Δ	Δ	Δ	X	X	X	Δ	X	X	X	Δ	Δ	X	X	X	Δ	4	X	2	X
shadow	X	X	Δ'	X	Δ'	Δ'	Δ'	X	Δ'	X	Δ'	Δ'	Δ'	Δ'	X	X	X	Δ'	Δ'	Δ'	Δ'	Δ'	X	X	Δ'	X	Δ'
focus	Δ	Δ	X	1	X	Δ	2'	Δ	1	Δ	X	X	X	X	X	X	X	4	4	X	Δ	4	Δ	Δ	X	X	1
defocus	Δ	Δ	X	1	X	Δ	2'	Δ	1	Δ	X	X	X	X	X	X	X	4	4	X	Δ	4	Δ	Δ	X	X	1
font changing	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	X	X	X	Δ	Δ	X	X	X	X	X	X	Δ	X
color changing	Δ	Δ	Δ	Δ	Δ	Δ	Δ	X	X	X	X	X	X	X	X	X	Δ	X	X	Δ	X	X	X	X	Δ	Δ	X
scaled-up	Δ	X	Δ	Δ	Δ	2'	Δ	X	2'	X	X	X	X	X	1'	Δ	X	X	X	Δ	Δ	X	1	X	X	X	Δ
scaled-down	Δ	X	Δ	Δ	Δ	2	Δ	X	2'	X	X	X	X	X	1'	Δ	X	X	X	Δ	Δ	X	1	X	X	X	Δ
transparency	X	X	Δ'	X	Δ'	X	X	X	Δ'	Δ'	Δ'	Δ'	Δ'	X	X	X	X	X	X	Δ'	X	Δ'	X	X	Δ'	Δ'	Δ'
scrolled	X	X	X	X	X	2'	X	Δ	X	Δ	Δ	Δ	Δ	X	X	2'	X	X	X	X	Δ	4'	Δ	X	X	X	X
toggle	Δ	1	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	3	4	X	X	X	1	X	X	Δ	X
text entered	Δ	X	2	X	X	2	2	Δ	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Δ	X	X	X
text deleted	Δ	X	Δ	X	X	2	2	Δ	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Δ	X	X	X
collapse	X	1	X	X	X	X	X	Δ	X	1	1	2'	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
uncollapse	X	1	X	X	X	X	X	Δ	X	1	1	2'	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
fade in	X	F	F	F	F	F	F	F	F	F	F	F	F	F	F	X	F	F	F	F	F	F	X	F	F	X	F
fade out	X	F	F	F	F	F	F	F	F	F	F	F	F	F	F	X	F	F	F	F	F	F	X	F	F	X	F
animation	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	F	X	X	X	X	F	X	X

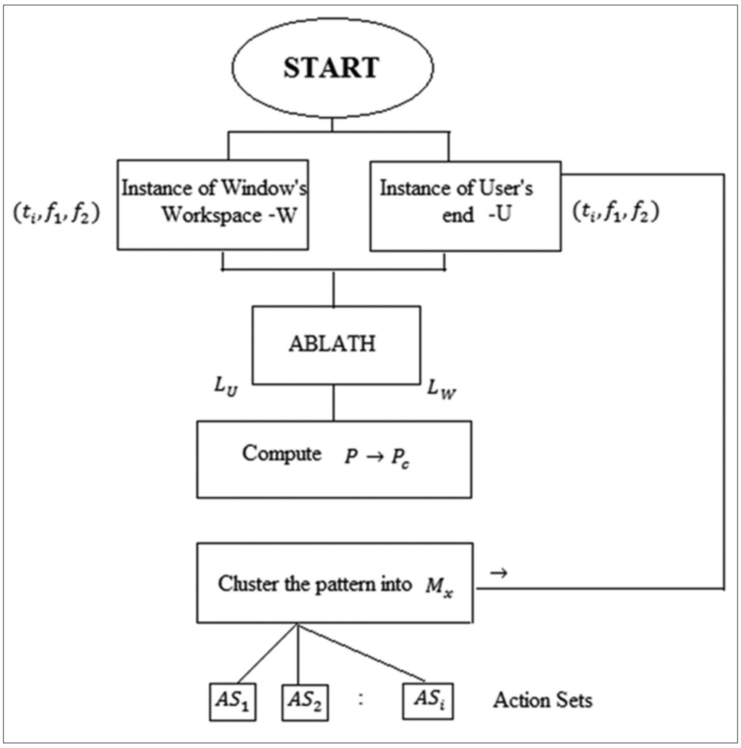


Fig. 1b: Flow chart of mobile evolutionary algorithm for evasive interactive graphical user interfaces

Where, AS_i is the automated classified action sets for example: Dragging, dropping, closing, etc.

Step 5: Repeat step 1-4, update P_c.

CONCLUSION

Fig. 1a gives the summarized the analysis of the performance result for the testable complex operations (Represented by triangle, and order of execution by numeric 1 and 2, respectively) performed over the proposed GUI algorithm and data flow process (Fig. 1b). Note: F is not testable operations. This paper presented and revealed a dynamically adaptive approach for automotive interaction with GUIs and offering various benefits and capabilities. We conclude by mentioning future offering by the proposed work.

Fig. 2: Sample augmented reality game controlled through the proposed technique

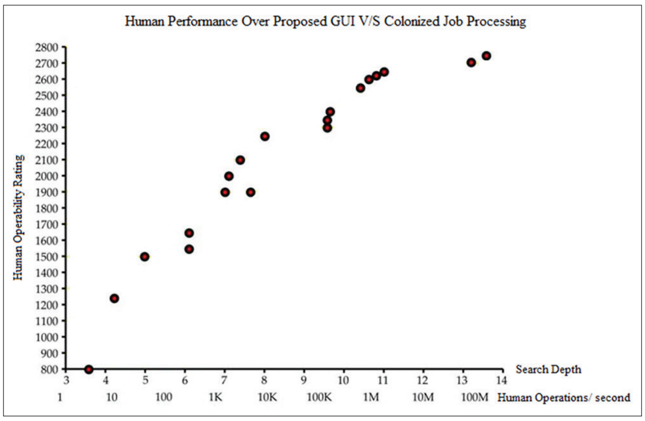


Fig. 3: Human performance test against the colonized job processing done over the proposed graphical user interface

Many users prefer a personalized experience we have successfully offered it with satisfaction and speed of performing tasks on computing device. This is the robust method to the present data and provides a advantageous utility to the users employed in environments in highly colonized work environments such as flight control and management unit, scientific laboratories, nuclear Power Stations and in Conferences or Academic/Business presentations. This automation is less affected by variations when users are entitled to work in crowded areas (as shown in Fig. 2). The operational performance is shown in Fig. 3. The users can save the template of the trained GUI and can forward it to their other mobile devices, such that this will be uninfluenced by theconversion function to map patterns between themes or to require users to normalize the execution environment by switching to the default workspace.

REFERENCES

1. Memon AM, Pollack ME, Soffa ML. Using a Goal-driven Approach to Generate Test Cases for GUIs. ICSE '99 Proceedings of the 21st International Conference on Software Engineering; 2002.
2. Clarke JM. Automated test generation from a behavioral model. In: Proceedings of Pacific Northwest Software Quality Conference. IEEE Press, May; 1998.
3. Esmelioglu S, Apfelbaum L. Automated test generation, execution and reporting. In: Proceedings of Pacific Northwest Software Quality Conference. IEEE Press, October; 1997.
4. Howe A, von Mayrhauser A, Mraz RT. Test case generation as an AI planning problem. Autom Softw Eng 1997;4:77-106.
5. Memon AM, Pollack ME, Soffa ML. Hierarchical GUI test case generation using automated planning. IEEE Trans Softw Eng 2001;27(2):144-55.
6. Koehler J, Nebel B, Hoffman J, Dimopoulos Y. Extending planning graphs to an ADL subset. Lect Notes Comput Sci 1997;1348:273.
7. Kepple LR. The black art of GUI testing. Dr. Dobb's J Softw Tools 1994;19(2):40.
8. Hammontree ML, Hendrickson JJ, Hensley BW. Integrated data capture and analysis tools for research and testing on graphical user interfaces. In: Bauersfeld P, Bennett ,Lynch G, editors. Proceedings of the Conference on Human Factors in Computing System. New York, NY, USA: ACM Press; 1992. p. 431-2.
9. Kasik DJ, George HG. Toward automatic generation of novice user test scripts. In: Tauber MJ, Bellotti V, Jeffries , Mackinlay JD, Nielsen J, editors. Proceedings of the Conference on Human Factors in Computing Systems: Common Ground. New York: ACM Press; 1996. p. 244-51.
10. Rai A. Attribute based level adaptive thresholding algorithm for object extraction. J Adv Robot 2014;1(1):29-33.
11. Rai A. Shell implementation of neural net over the UNIX environment for file management: A step towards automated operating system. J Oper Syst Dev Trends 2014;1(2):10-4.
12. Rai A. Air computing: A parallel computing module for offloading computational workload on neighboring android devices. Recent Trends Parallel Comput 2015;1(3):10-3.
13. Rai A, Sakkaravarthi R. Distributed learning in networked controlled cyber physical system. Int J Pharm Technol 2016;8(3):18537-46.