RIGHT MOUSE BUTTON SURROGATE ON TOUCH SCREENS

Jan Vanek and Bruno Jezek
Faculty of Military Health Sciences, University of Defence, Trebesska 1575, 50002 Hradec Kralove, Czech Republic

Keywords: Right mouse button, Touch screen, Gesture, Statistics.

Abstract: The pervasiveness of computer systems is largely determined by their ease of use. Touch screens have proven to be a natural interface with a strong sensorimotor feedback. Although multi-touch technologies are ever more popular, single-touch screens are still often preferred. They are cheaper and they map directly to pointing devices such as computer mice, thus requiring no software modifications. Therefore, they can easily be integrated into existing systems with WIMP interfaces, e.g. MS Windows based systems. Applications in such systems often rely on user pressing the right mouse button to open context menus etc. Since single-touch screens register only one touch at a time, different methods are being used to allow a user to determine the outcome of the touch. The paper proposes a new interaction scheme for this purpose and an algorithm to detect it.

1 INTRODUCTION

A significant performance increase and new discoveries in the field of computer technologies and electronics bring great advancements also in the area of human-computer interaction (HCI). New and old user interface modalities, which complement the long established and widespread WIMP interface (Windows, Icons, Menus, Pointing device) (Anthes, 2008) (Taylor, 1997), are being employed in real life. If implemented correctly, multimodal interaction can improve user interface efficiency and lead to greater work effectivity (Raisamo, 1999).

Devices that combine a display unit with a touch sensor are a rediscovered input modality, which, thanks also to its commercial availability, is gaining momentum in the field of HCI. All such devices will be referred to as touch screens in the following text. Touch screens convey a direct bond between displayed information and haptic interaction and thus are very intuitive and close to lowest level sensorimotor processes, which has manifested not only in human psychology research (for example (McGuire et al., 2000), (Weber et al., 2003) and others), but also in animal behaviour experiments (for example (Conway & Christiansen, 2001), (Hashiya & Kojima, 1997) and many others). The popularity of touch screens is also due to their hardware, software and functional compatibility with computer mice, which significantly simplifies their application to areas, where information technology with typical user interfaces is already being extensively used.

Today, common WIMP user interfaces depend in much functionality (context menu, extended object manipulation) on input from an at least two button mouse. However, most of the touch screen technologies allow only one touch to be registered. This article proposes a new method of right mouse button press emulation on touch screens for some touch screen technologies and its implementation.

2 TOUCH SCREENS

Although touch screens got major commercial attention in past several years their history is rather long. The first touch screen was developed by Sam Hurst in Elographics in 1971 (Ellis, 2007), the first personal computer equipped with a touch screen was launched to market by Hewlet Packard in 1983 (Knight, 2007). Since that time touch screens have found a wide range of application including information kiosks, register desk systems, educational presentation boards and a large class of mobile digital devices.

Several technologies and physical principles are used to detect touch. Resistive and capacitive touch screens are the most widespread, as they are robust, adequately sensitive and have low production costs.

Vanek J. and Jezek B..
RIGHT MOUSE BUTTON SURROGATE ON TOUCH SCREENS.
DOI: 10.5220/0003504803040309
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 304-309
ISBN: 978-989-8425-56-0

Other technologies include strain gauging, surface acoustic wave, optical imaging and frustrated total internal reflection.

In many touch screen usage cases individual applications have their user interface tailored to the given task and touch interaction. Especially technologies that register multiple simultaneous touches and thus cannot have their output mapped to a pointing device require user interfaces to be designed with respect to that (Buxton, 2011) (Nichols, 2007). However, touch screens are often employed as a mouse compatible complement to a personal computer with general, unspecified use. This includes already widespread presentation and education screens such as SmartBoard, but also special industrial solutions, such as the Interactive Mural touch wall (Guimbretière, Stone, & Winograd, 2001). Such kind of a setup then runs a variety of applications in a WIMP user interface of some operating system, usually MS Windows.

As mentioned in the introduction, many user activities in a WIMP interface involve a right mouse button press. The most prevalent touch screen technologies, however, interpret touch only as a single left mouse button press. Therefore, different methods are being used to emulate the missing right mouse button on touch screens in general applications.

3 RIGHT MOUSE BUTTON SURROGATE

Basically two methods are being used to surrogate the right mouse button on touch screens. Following the first method, usually called tap&hold, a user maintains touch for at least the set time period, after which the touch is interpreted as a right mouse button pressed and released event. A short time period can cause frequent false alarms, i.e. incorrect touch interpretation as a right mouse button press user did not intend. On the other hand, a longer time period increases the delay before following activities and also amplifies motoric strain, especially on large touch boards.

Second class of methods requires the user to notify the system before the intended right mouse button press. Individual implementations can differ; some manufacturers solve the problem at the software level – user touches some object displayed on screen; other solutions include physical button placed near the touch screen. This method, even more than tap&hold, hinders users in their work and causes grater strain from repetitive movements. This is prominent especially on large touch boards in combination with a physical button.

From the construction of resistive touch screens it is possible to infer that multiple simultaneous touches will be interpreted as a single touch in the centre of pressure. This assumption was empirically verified on several resistive and capacitive technology devices from different manufacturers.

The assumption leads to the new right mouse button substitution method proposed in the paper. The intended right mouse button press is indicated by sequentially simultaneous touch of two fingers. The user touches the desired point with her index finger, maintains the touch and simultaneously presses and releases her middle finger in adequate distance and, finally, releases her index finger. The first touch sets coordinates to the desired point. The increasing pressure from the second finger causes the coordinates to move towards the point of the second touch. The movement stops when the pressure is maximal and after that coordinates move back to the desired point. See figure 1. After the first finger is released, respectively after coordinates reach the desired point, the touch is interpreted as a right mouse button click, respectively as a right mouse button press.

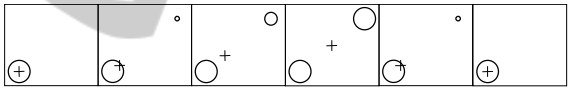

Figure 1: Coordinates movement caused by second touch, pressure illustrated by circle diameters.

The described semantic assignment of the index and middle finger is natural and intuitive (the fingers usually operate left and right mouse button), but the proposed method does not depend on it; any two fingers of one or both hands in any order can be used. The important thing is the sequence press first – press second – release second – release first.

The proposed interaction scheme may be detected directly in hardware based on the used technology, but it can also be done based solely on the inferred coordinates.

3.1 Implementation

Since the proposed interaction scheme can be evaluated from coordinates, the method can be implemented in software independently of the touch screen hardware. The captured coordinates, interpreted by the operating system as mouse coordinates, change in time when second touch occurs. The resulting path, which is sampled into

linear segments, can be considered a single-stroke gesture and as such it can be processed and evaluated.

There are a number of different approaches to gesture recognition. Some authors use purely geometrical algorithms (Hammond & Davis, 2006) (Wobbrock, A. D. Wilson, & Li, 2007), some construct ad-hoc algorithms (Notowidigdo & Miller, 2004) (A. Wilson & Shafer, 2003), sometimes based on very simple principles (Lank, Thorley, & Chen, 2000); other approaches employ image analysis (Kara, 2004) or neural networks (Pittman, 1991). Rather complicated methods use dynamic programming (Myers & Rabiner, 1981) (Tappert, 1982) or hidden Markov models (Anderson, Bailey, & Skubic, 2004) (Cao & Balakrishnan, 2005) (Sezgin & Davis, 2005). Probably the most cited works are based on statistical feature classification (Cho, Oh, & Lee, 2004) (Cho, 2006), with the best known being (Rubine, 1991).

The gesture resulting from the second touch can certainly be detected using some of the existing gesture recognition algorithms. However, the general algorithms are not optimal for this specific task. They are used mostly in environments, where gestures are initiated modally with a predefined indication. Therefore, the algorithms are built on the premise that the input path is a gesture and thus only solve the task of differentiation between known gestures. The problem of gesture recognition within a set of movements with varying user intentions is in most of the algorithms considered only marginally, if at all. The proposed method, however, requires evaluation of movements for which it is not known whether the user is attempting to perform the gesture or not. The implementation of the method must facilitate this detection.

Because the gesture attempt is indicated in advance, most of the algorithms work with the complete gesture. That does not allow for right mouse button drag&drop operation using the proposed interaction scheme. In order to facilitate on-the-fly movement processing the proposed method implementation must allow for sequential mouse coordinates path evaluation with constant computational complexity per segment.

For these reasons we have designed an algorithm optimized for the described gesture, which is computationally efficient and can be applied on-the-fly with constant time with respect to the number of already processed segments.

3.2 The Algorithm

The algorithm was designed based on data acquired on a for-wire resistive touch screen ADI V-Touch 1710. The driver of the display maps touch to system mouse coordinates. To record the data we created software that with frequency of 64Hz using DirectInput interface in non-exclusive background mode captures mouse coordinates. The data was recorded per individual complete gestures derived from touch press, movement and release. Each gesture vertex is described by absolute screen coordinates in pixels and time in milliseconds that passed since the previous vertex was recorded.

400 reference gestures were recorded by several users. The gestures were performed to follow the proposed interaction scheme. The gestures originated at randomly selected locations distributed across the whole screen so that the influence of eventual local touch screen irregularities was eliminated. The same set of locations was used by all the users.

Figure 2 shows the first and third reference gesture plotted in absolute pixel coordinates. Segment vertices are complemented with time in milliseconds that passed since the start of the gesture. It is clear that the algorithms that use vertex data directly are not suitable for the task. Gestures do not correspond in either number of segments, screen-space location, size, rotation or even in topology.

A new variable, which is invariant to rigid affine transformations, can be derived from the screen coordinates. Let us denote the variable by C. Let us set $c_1 = c_2 = 1$. For $i = 3,\ldots, n$ let us set

$$c_i = \frac{(x_i - x_{i-1})(x_{i-1} - x_{i-2}) + (y_i - y_{i-1})(y_{i-1} - y_{i-2})}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}\sqrt{(x_{i-1} - x_{i-2})^2 + (y_{i-1} - y_{i-2})^2}} \quad (1)$$

In other words, $C = \cos(\Phi)$, where φ_i is the angle between direction vectors of i-1 and i-2 segments (see figure 3) and n is the number of gesture segment vertices.

In order to compare a gesture to the reference ones, gestures have to be described by a fixed set of attributes. The reference gestures exhibit several characteristic features, as figure 2 suggests. That can be taken advantage of and each gesture can be described by a vector of the characteristic features.

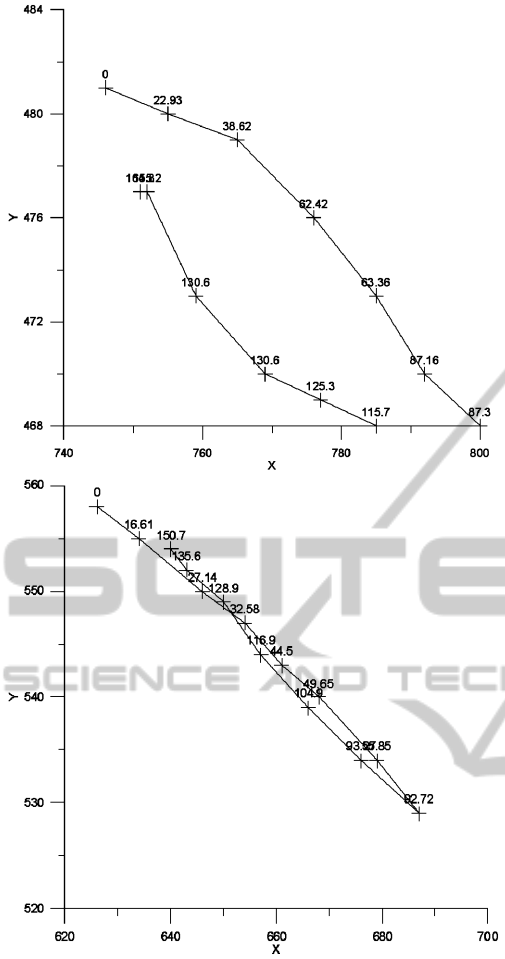

Figure 2: Example of reference gesture plots at absolute pixel coordinates.

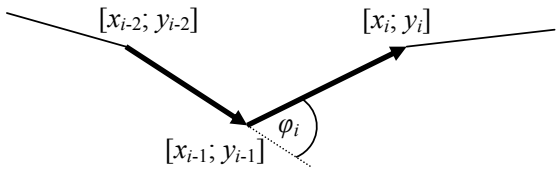

Figure 3: Angle between two segments.

The most prominent feature of the reference gestures is exactly one occurrence of point of return, i.e. exactly one negative value of variable *C*. Each gesture with exactly one point of return is then described by selected characteristic features in the vector (N, T_O, L_O, $\overline{|C|}$, min($|C|$)), where N is the number of vertices of gesture segments, T_O is the normalized time of point of return, L_O is the normalized length at the point of return, $\overline{|C|}$ is the average of absolute values of variable *C* and min($|C|$) is the minimum of absolute values of variable *C*. Table 1 shows vectors of sample means, standard deviations and coefficients of variation of the selected features of the reference gestures. The coefficients of variation suggest that the selected features describe the reference gestures tightly enough.

Table 1: Sample statistics of the selected features of the reference gestures.

Statistic	N	T_O	L_O	$\overline{\lvert C \rvert}$	min($\lvert C \rvert$)
$\bar{x}$	12.306	0.731	0.647	0.986	0.907
s	3.244	0.119	0.088	0.019	0.151
CV	0.264	0.162	0.135	0.020	0.166

The main output of the proposed algorithm is the decision whether the input gesture should be interpreted as a result of the proposed touch scheme or not, i.e. as user's indication of an intended right mouse button press. The vector of sample means of the reference gestures will serve as an etalon, to which input gestures are compared using a suitable distance metric. Mahalanobis distance was used, as it takes into account the differences in feature variances and the correlations of features.

To determine a suitable distance threshold all mouse movements with the left mouse button pressed were recorded on several users' computers within the course of one working day. These gestures are equivalent to gestures that do not result from the proposed interaction scheme, i.e. all the touches that are not intended as a right mouse button press following the proposed method. These movements will be referenced as random gestures in the following text. 4052 gestures were recorded in total.

The distance threshold is selected so that both the probability of intended gesture rejection and the probability of random gesture acceptance are reasonably low. Table 2 shows an example of distance thresholds, out of which the threshold of 6 gives the best results for the recorded data. This way the numbers of incorrectly evaluated gestures reach circa 2% of the recorded gestures in each category. The optimal threshold can be calculated by minimizing the disparity between the percentages of incorrectly classified reference and random gestures, but such precision would be superfluous considering the fact that each user may prefer different recognition sensitivity.

Table 2: The numbers of reference gestures with distances from etalon exceeding example thresholds and random gestures with exactly one point of return and with distances below the thresholds.

Threshold	D_{Ref} > threshold	$D_{Rand} \leq$ threshold
2	175	1
4	36	22
6	8	101
8	2	190
10	1	258
12	0	326

4 RESULTS AND CONCLUSIONS

The algorithm was implemented as a prototype using DirectInput API to read coordinates in background. The resulting application was tested on a resistive touch display ADI V-Touch 1710, capacitive touch display NEC V-Touch 1921 CU and resistive touch wall SmartBoard 540. Computational load was immeasurable on all the computer setups. Gesture detection error rate and its dependence on the distance threshold corresponded to expectations. The touch wall produced high noise in the recorded data, which caused frequent points of return and gesture rejection with segment lengths close to one or two pixels. To remove the noise a segment was recorded only after it reached a defined minimal length. The minimal length of four pixels yielded results equivalent to those on the displays.

To assess the efficiency of the proposed method of right mouse button surrogate and its comparison to the tap&hold method, software button method and hardware button method an experiment was designed, in which users react to a series of graphical symbols with either left or right virtual mouse button press in dependence on the currently displayed symbol. Expected type of reaction, reaction time and the number of corrections are recorded in the course of the task.

For technical, organizational and economic reasons the experiment is yet to be performed on a statistically significant sample of users. Thorough analysis of the method impact on user performance thus remains future work. However, preliminary tests taken by a limited number of users suggest the potential of the method especially in comparison with the button methods. The tests also show that the method requires some dexterity and practice, but that the touch interaction scheme is intuitive.

The proposed method of right mouse button surrogate on touch screens is a sound alternative to other existing methods in use, but it is not intended as a complete replacement. The described detection algorithm of the proposed touch interaction scheme is computationally efficient and easy to implement and therefore it can be integrated both into the software driver and the firmware of the underlying touch screen hardware.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Defence research project MO0 FVZ0000604.

REFERENCES

Anderson, D., Bailey, C., & Skubic, M. (2004). Markov Model Symbol Recognition for Sketch-Based Interfaces. *Proceedings of AAAI Fall Symposium* (pp. 15-21). Presented at the AAAI Fall Symposium, Menlo Park, CA: AAAI Press.

Anthes, G. (2008). Give your computer the finger: Touch-screen tech comes of age - Computerworld. Retrieved February 16, 2011, from http://www.computerworld.com/s/article/9058841/Give_your_computer_the_finger_Touch_screen_tech_comes_of_age

Buxton, B. (2011). Multi-Touch Systems that I Have Known and Loved. *Multi-Touch Systems that I Have Known and Loved*. Retrieved February 16, 2011, from http://www.billbuxton.com/multitouchOverview.html

Cao, X., & Balakrishnan, R. (2005). Evaluation of an on-line adaptive gesture interface with command prediction. *Proceedings of Graphics Interface 2005* (pp. 187-194). Victoria, British Columbia: Canadian Human-Computer Communications Society.

Conway, C. M., & Christiansen, M. H. (2001). Sequential learning in non-human primates. *Trends in Cognitive Sciences*, *5*(12), 539-546.

Ellis, N. (2007). Sam Hurst Touches on a Few Great Ideas. *Berea College Magazine*, *77*(4), 22-27.

Guimbretière, F., Stone, M., & Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01 (p. 21–30). New York, NY, USA: ACM.

Hammond, T., & Davis, R. (2006). Tahuti: a geometrical sketch recognition system for UML class diagrams. *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06. New York, NY, USA: ACM.

Hashiya, K., & Kojima, S. (1997). Auditory-visual Intermodel Matching by a Chimpanzee (Pan troglodytes). *Japanese Psychological Research*, *39*(3), 182-190.

Cho, M. G. (2006). A new gesture recognition algorithm and segmentation method of Korean scripts for gesture-allowed ink editor. *Information Sciences*, *176*(9), 1290-1303.

Cho, M. G., Oh, A. S., & Lee, B. K. (2004). A Feature-Based Algorithm for Recognizing Gestures on Portable Computers. *Computational Science and Its Applications – ICCSA 2004*, Lecture Notes in Computer Science (Vol. 3043, pp. 33-40). Springer Berlin / Heidelberg.

Kara, L. B. (2004). An Image-Based Trainable Symbol Recognizer for Sketch-Based Interfaces. *Proceedings of AAAI Fall Symposium* (Vol. 2, pp. 99-105). Presented at the AAAI Fall Symposium, Menlo Park, CA: AAAI Press.

Knight, M. (2007). In touch with the digital age. Retrieved February 16, 2011, from http://edition.cnn.com/2007/TECH/07/27/fs.touchscreen/index.html

Lank, E., Thorley, J. S., & Chen, S. J.-S. (2000). An interactive system for recognizing hand drawn UML diagrams. *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research* (p. 7). Mississauga, Ontario, Canada: IBM Press.

McGuire, M., Bakst, K., Fairbanks, L., McGuire, M., Sachinvala, N., von Scotti, H., & Brown, N. (2000). Cognitive, mood, and functional evaluations using touchscreen technology. *The Journal of Nervous and Mental Disease*, *188*(12), 813-817.

Myers, C. S., & Rabiner, L. R. (1981). A Comparative Study Of Several Dynamic Time-Warping Algorithms For Connected Word Recognition. *The Bell System Technical Journal*, *60*(7), 1389-1409.

Nichols, S. J. V. (2007). New Interfaces at the Touch of a Fingertip. *Computer*, *40*, 12–15.

Notowidigdo, M., & Miller, R. C. (2004). Off-Line Sketch Interpretation. *Proceedings of AAAI Fall Symposium* (Vol. 2, pp. 120-126). Presented at the AAAI Fall Symposium, Menlo Park, CA: AAAI Press.

Pittman, J. A. (1991). Recognizing handwritten text. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91 (p. 271–275). New York, NY, USA: ACM.

Raisamo, R. (1999). *Multimodal Human-Computer Interaction: a Constructive and Empirical Study* (Dissertation). University of Tampere, Tampere.

Rubine, D. (1991). Specifying gestures by example. *ACM SIGGRAPH Computer Graphics* (Vol. 25, p. 329–337). New York, NY, USA: ACM.

Sezgin, T. M., & Davis, R. (2005). HMM-based efficient sketch recognition. *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05 (p. 281–283). New York, NY, USA: ACM.

Tappert, C. C. (1982). Cursive script recognition by elastic matching. *IBM Journal of Research and Development*, *26*, 765–771.

Taylor, A. G. (1997). *WIMP Interfaces* (Topic Report No. CS6751). Winter '97. Atlanta, GA: Georgia Tech. Retrieved from http://www.cc.gatech.edu/classes/cs6751_97_winter/Topics/dialog-wimp/

Weber, B., Schneider, B., Fritze, J., Gille, B., Hornung, S., Kühner, T., & Maurer, K. (2003). Acceptance of computerized compared to paper-and-pencil assessment in psychiatric inpatients. *Computers in Human Behavior*, *19*(1), 81-93.

Wilson, A., & Shafer, S. (2003). XWand: UI for intelligent spaces. *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03 (p. 545–552). New York, NY, USA: ACM.

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07 (p. 159–168). New York, NY, USA: ACM.