Undo/Redo by Trajectory

Tatsuhito Oe, Buntarou Shizuki, and Jiro Tanaka

University of Tsukuba, Japan
{tatsuhito,shizuki,jiro}@iplab.cs.tsukuba.ac.jp

Abstract. We have developed a trajectory based undo/redo interface. Using the interface, a user traces actions' trajectories shown on a display. As a result, undo/redo manipulations are performed rapidly with selection of a target from a history. In this paper, we describe interaction techniques, implementation, and advanced usages of the interface.

Keywords: undo/redo, trajectory, history, tracing, desktop interface, gui.

1 Introduction

In many applications, user's actions (e.g., key commands or mouse actions) are stored as history for undo/redo (Figure 1a). When the user undoes/redoes these actions, the user executes commands for undo/redo one or more times (Figure 1b). Because the user must execute commands several times, performing undo/redo is time-consuming.

To perform undo/redo manipulation faster, the following approaches have been developed that allow the user to undo/redo actions selectively.

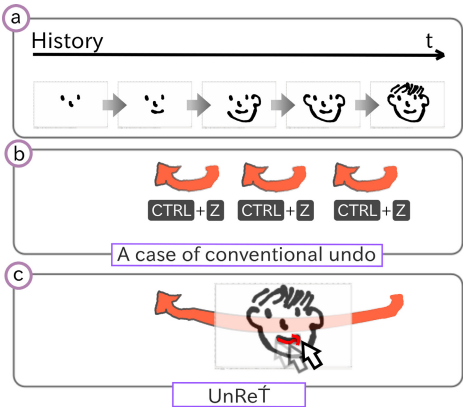

Fig. 1. Conventional undo and UnReŤ's undo in linear history model

M. Kurosu (Ed.): Human-Computer Interaction, Part IV, HCII 2013, LNCS 8007, pp. 712–721, 2013.

Some approaches visualized a history using texts or screenshots. For example LibreOffice Impress[1] showed a list of manipulations as texts, and Meng et al. [13] presented a visualization of selective undo [6] using screenshots.

Other approaches localized undo/redo manipulation's region. For example, regional undo model [9] enables the user to undo/redo actions that occur in a specific region of a display. Moreover, a selective undo model [6] enables the user to undo/redo an isolated action.

These two approaches have an advantage and a disadvantage. The former approach enables a user to look an overview of a history rapidly, though takes a time to undo/redo if the history becomes large. By contrast, the latter enables the user to undo/redo rapidly even when the history is large, though it cannot be applied to the linear history model used in most applications.

Our goal is to explore the effectiveness of an undo/redo interface where a user can undo/redo by selecting a manipulation directly. To achieve our goal, we developed a trajectory based undo/redo interface (*UnReT́*[2]). Using UnReT́, the user can undo/redo by tracing a mouse trajectory (one trajectory consists of mouse press, drag, and release) as shown in Figure 1c. Because the user does not need to execute many commands, the user can perform undo/redo manipulation faster.

2 Related Work

UnReT́ is an application-independent interface implemented by extending a desktop environment in which a user undoes/redoes actions by tracing a past trajectory visualized on the desktop, even on an application employing the linear history model. Therefore, related research into the interface includes work on history visualization using texts or screenshots, a history model, and a desktop extension.

2.1 History Visualization Using Texts or Screenshots

Some research has tried to visualize a history using texts or screenshots. In this research area, simple visualization uses a list of texts or screenshots. For example, Meng et al. [13] visualized the history using a list of screenshots.

In contrast to the simple visualization, some research has tried to represent additional information over texts or screenshots. Kurlander et al. [12] visualized actions' context using pairs of two screenshots before and after an action. Nakamura et al. [15] overlaid GUI actions on a screenshot to improve search speed from the history. In addition, Vratislav [19] adopted Fisheye Menus [5] to the texts' list of the history to improve search speed.

In this research, a user undoes/redoes actions by viewing and selecting an action from the list. By contrast, in UnReT́, the user undoes/redoes by tracing a trajectory shown on a display. Therefore, using UnReT́, the user does not need to look at the list, so the user can undo/redo faster.

[1] `http://www.libreoffice.org/`

[2] UnReT́ is short for "**Un**do/**Re**do by **T**rajectory".

2.2 History Model

A history model has been researched that localizes undo/redo's manipulation region spatially.

Berlage [6] and Myers et al. [14] presented selective undo model that enables a user to undo/redo an isolated action. Kawasaki et al. [9] presented regional undo model that has a broader region for undo/redo manipulation than the selective undo model. These history models have features to localize manipulation's region spatially. Therefore, these models can be applied not only to a single user's environment but also to a multiple users' environment where one user performs undo/redo actions in his/her own region [17,18].

These history models cannot be applied to a general application based on the linear history model. On the other hand, UnReT́ can be applied to various applications based on the linear model, because UnReT́'s implementation is independent from a particular application. That is, the implementation is based on mouse manipulations' trajectories stored in a desktop environment.

2.3 Desktop Extension

UnReT́ is the interface that extends normal undo/redo manipulation by capturing mouse trajectory. Related to UnReT́, there is research on extending history or mouse manipulation in a desktop environment.

Extending history. Interfaces have been researched that assist a user by capturing his/her actions or states in a desktop environment. For example, Rekimoto [16] showed Time-Machine Computing that can recover any past state in the desktop. In addition, Kelly et al. [10] presented Desktop History that captured actions to visualize files manipulated at any application. Furthermore, Grossman et al. [8] captured actions and videos in the desktop to allow the user to watch the videos at any past action.

In these interfaces, captured actions are used for presenting a user's past manipulations. By contrast, captured actions are used for undo/redo manipulation in UnReT́.

Extending mouse manipulation. There has been research on extending a user's mouse manipulation in a desktop environment. Appert et al. [3] developed Dwell-and-Spring, which enables a user to undo and cancel mouse manipulation on the basis of a spring metaphor between a cursor and a target. Kobayashi et al. [11] presented Boomerang, which allows the user to move files between directories by using throwing gesture. Similar to the above research, UnReT́ extends undo/redo manipulation also by using mouse manipulation in the desktop.

3 Undo/Redo by Trajectory

UnReT́ is an interface where a user can undo/redo by tracing a past mouse trajectory. Below is how to undo/redo in UnReT́.

Undo: The user holds down the Ctrl key and traces a trajectory with a pointer. As a result, past actions including the one that gave the trajectory are undone (Figure 2a).

Redo: Holding a Ctrl key for long enough visualizes possible target trajectories. Thereafter, the user traces a trajectory. As a result, redo is performed (Figure 2b).

To support these manipulations, the system provides the functions below.

Preview of undo/redo. As shown in Figure 2c, the user can perform *Scratch gesture* while tracing the trajectory. This shows preview screenshots of undo/redo targets. Thereafter the user can undo/redo by selecting one of the screenshots and also cancel undo/redo manipulation with the "Cancel" button shown in Figure 2c.

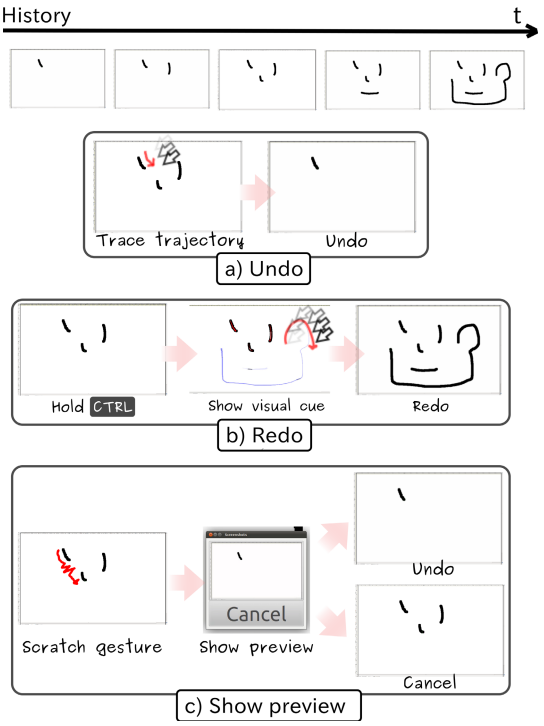

Fig. 2. Undo/redo interactions using UnReT́

Trajectory visualization. When the user holds down the Ctrl key for long enough, mouse trajectories are visualized (Figure 3a). Trajectories are red for an undo target and blue for a redo target (Figure 3b). This visualization helps the user to know where to trace.

Showing a list of screenshots when tracing *similar trajectories*. Similar trajectories occur especially in contour drawing (Figure 4a). When the user traces such trajectories, applicable targets are shown as the list of screenshots (Figure 4b), and then the user selects an item from the list for performing undo/redo.

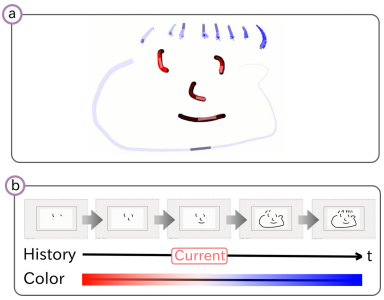

Fig. 3. Showing visual by holding down Ctrl key for long enough

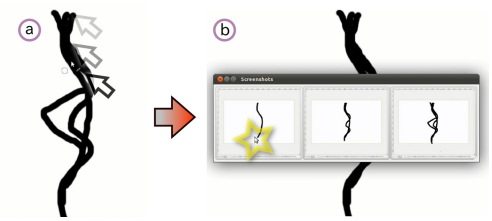

Fig. 4. Showing the list of screenshots when similar trajectories are traced

4 Implementation

Implementation of UnReŤ consists of mouse trajectory recording (Figure 5a), mouse trajectory matching (Figure 5b), and undo/redo processing (Figure 5c).

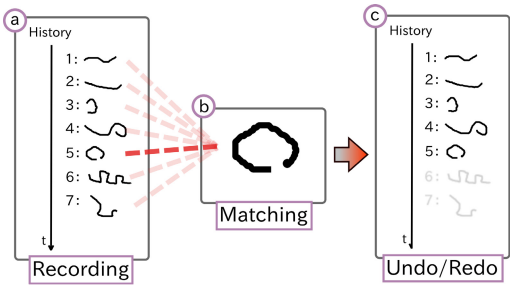

Fig. 5. UnReŤ consists of recoding, matching, and undo/redo processing

4.1 Mouse Trajectory Recording

When a trajectory is input, the trajectory's absolute points on a display are recorded to a history (Figure 6).

4.2 Mouse Trajectory Matching

When a user traces a trajectory while holding down the Ctrl key, similarities between the input trajectory and trajectories in the history are calculated using *Dynamic Programming (DP) Matching* (Figure 7). Then the system undoes actions until the trajectory with the highest similarity is input.

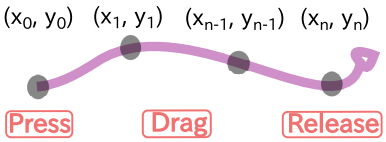

Fig. 6. Mouse trajectory recording

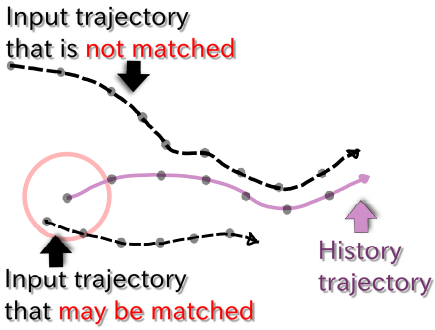

Fig. 7. Mouse trajectory matching

4.3 Undo/Redo Processing

In undo/redo processing, the system sends key shortcuts multiple times to the application. Because key shortcuts differ for each application, we implemented a function that enables a user to register undo/redo processing below.

undo/redo processing by key shortcuts. The system sends a registered key shortcut.

undo/redo processing by *reversed mouse manipulation.* Reversed mouse manipulation means inputting mouse's release, drag, and press from a trajectory in the history. In this process, Figure 6's (x_n, y_n) becomes the press point, (x_{n-1}, y_{n-1}),...,(x_1, y_1) become drag points, and finally (x_0, y_0) becomes the release point.

Using the register function, a user can register a process such as "If an application is GIMP, the system sends Ctrl+z key event".

5 Applications

We applied UnReT́ to various environments and manipulations to explore UnReT́'s effectiveness. In this section, we describe trial results.

5.1 Environments

We tested UnReŤ in environments using a mouse and a keyboard, a stylus interface, and a touch interface, as shown in Figure 8. In every environment, UnReŤ was used by the first author using a GIMP application. Each environment's trial results are shown below.

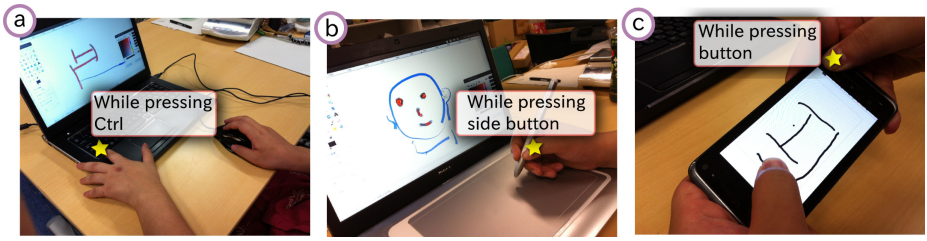

Fig. 8. Environments where UnReŤ was tested

Mouse and keyboard (Figure 8a). We used UnReŤ by using the mouse with the right hand while pressing the Ctrl key with the left hand (Figure 8a⋆). As a result, we observed that UnReŤ worked well in the environment. However, when the user needed to undo/redo manipulation only several times, inputting key shortcuts was more effective than UnReŤ.

Stylus interface (Figure 8b). We used UnReŤ with the stylus whose side button (Figure 8b⋆) was assigned as the Ctrl key. The first author did not use the stylus interface usually. Nevertheless, using the stylus made it easier to perform UnReŤ than using the mouse and the keyboard. The reason is that the stylus is a more suitable interface to trace a trajectory than the mouse. This result was consistent with that of Accot et al. [2], which revealed that the stylus performs better than the mouse for tracing by evaluating tracing performance of input devices based on Steering Law [1].

Touch interface (Figure 8c). We used UnReŤ on a mobile device. In this environment, we traced a trajectory with the left hand while holding down the mobile device's button with the other hand (Figure 8c⋆). This environment was realized by accessing a remote Linux server running our system through a VNC client on the mobile device. As a result, using the touch interface enabled the user to trace the trajectory directly. This result could be applied to various touch interfaces.

5.2 Manipulations

We tested UnReŤ with various manipulations. Similar to Section 5.1, UnReŤ was used by the first author in every manipulation.

Drawing manipulation. We used UnReŤ for drawing manipulation using GIMP. As a result, we found a technique that enabled a user to perform

undo/redo manipulation before *changing a color*. The technique is performed as follows. At first, a user draws something (Figure 9a). Next, the user changes the color (Figure 9b) and then draws (Figure 9c). After that, the user can look at the trajectory (Figure 9d⋆) when the user presses the Ctrl key for longer. Finally, the user can undo actions before changing the color by tracing the trajectory (Figure 9e).

We demonstrated drawing manipulation at a seminar of our university. At the seminar, we obtained comments such as "I want to undo/redo only an isolated action I trace", "Is there any idea of how to apply UnReT́ to other history models?".

For meet these requests, we need to adapt the selective undo model [6] to UnReT́. To implement selective undo in UnReT́, we plan to use the script undo model [4]. In the script undo model, if there is a sequence of actions $(A_1, ..., A_n)$, a user can undo/redo an isolated action $(A_i (1 \leq i \leq n))$ like in the selective undo model. This is done by undoing $A_n, ..., A_{i+1}, A_i$ normally and then restoring $A_{i+1}, ..., A_n$ using a script. By restoring recorded trajectories as the script, we can implement selective undo in UnReT́.

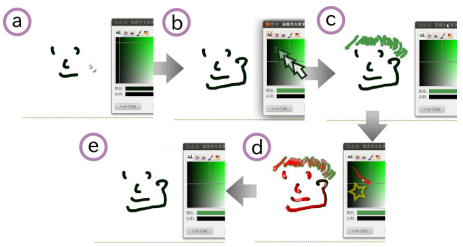

Fig. 9. Undo/redo before *changing a color*, which is typically impossible in GIMP

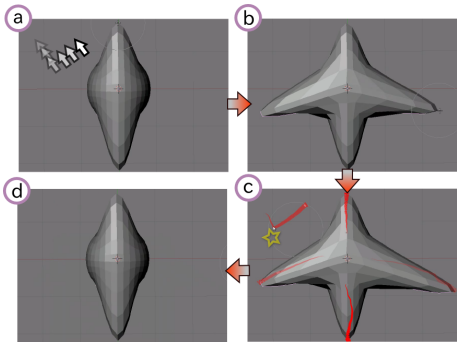

Fig. 10. A user defines a mark for undo/redo

Marking manipulation. Marking manipulation enables a user to define shortcuts for GUI manipulations like UIMarks [7]. Figure 10 illustrates how to apply marking manipulation to Blender[3] 3D modeler. In this example, first the user performs marking (Figure 10a). Then the user deforms the 3D model (Figure 10b). After that the user looks the trajectory of marking by holding down the Ctrl key for longer (Figure 10c⋆). Finally, the user traces the trajectory and undo/redo the action until the marking (Figure 10d).

Icon manipulation. Using the reversed mouse manipulation as undo/redo processing, we enable a user to undo/redo icon movements, which is impossible in a typical desktop environment. In Figure 11, first the user moves the icon

[3] `http://blender.jp/`

(Figure 11a). Then, the user looks at the icon movements' trajectories by pressing the Ctrl key for longer (Figure 11b), After that the user undoes icon movements by tracing the trajectory (Figure 11c). Similar to UnReŤ, icon movements were automated by Sikuli [20], using a programming by example of screenshots. In contrast to Sikuli, UnReŤ enabled the user to undo/redo icon movements.

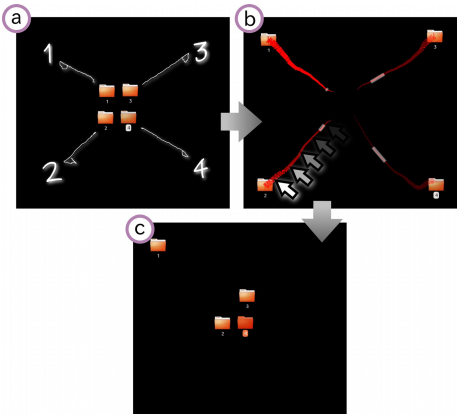

Fig. 11. Undo/redo icon movements

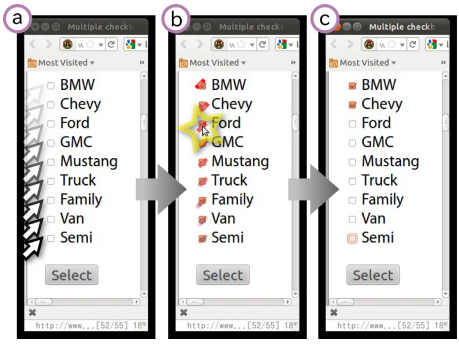

Fig. 12. Undo/redo checkbox selections

Checkbox manipulation. Similar to icon manipulation, a user can undo/redo selected checkboxes using the reversed mouse manipulation. In Figure 12, first the user selects checkboxes from top to bottom (Figure 12a). After that, the user clicks "Ford" (Figure 12b) while pressing the Ctrl key and then undoes selected checkboxes (Figure 12c).

6 Conclusion and Future Work

In this paper, we presented "Undo/Redo by Trajectory (UnReŤ)" and interaction techniques using UnReŤ. We applied UnReŤ to various environments and manipulations to explore effectiveness of the interface. Our future work is to implement selective undo manipulation in UnReŤ for further exploration.

References

1. Accot, J., Zhai, S.: Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks. In: Proc. CHI EA 1997, pp. 250–250. ACM (1997)
2. Accot, J., Zhai, S.: Performance Evaluation of Input Devices in Trajectory-based Tasks: An Application of The Steering Law. In: Proc. CHI 1999, pp. 466–472. ACM (1999)
3. Appert, C., Chapuis, O., Pietriga, E.: Dwell-and-Spring: Undo for Direct Manipulation. In: Proc. CHI 2012, pp. 1957–1966. ACM (2012)

4. Archer Jr., J.E., Conway, R., Schneider, F.B.: User Recovery and Reversal in Interactive Systems. ACM Transactions on Programming Languages and Systems (TOPLAS) 6(1), 1–19 (1984)
5. Bederson, B.B.: Fisheye menus. In: Proc. UIST 2000, pp. 217–225. ACM (2000)
6. Berlage, T.: A Selective Undo Mechanism for Graphical User Interfaces Based on Command Objects. ACM Transactions on Computer-Human Interaction 1(3), 269–294 (1994)
7. Chapuis, O., Roussel, N.: UIMarks: Quick Graphical Interaction with Specific Targets. In: Proc. UIST 2010, pp. 173–182. ACM (2010)
8. Grossman, T., Matejka, J., Fitzmaurice, G.: Chronicle: Capture, Exploration, and Playback of Document Workflow Histories. In: Proc. UIST 2010, pp. 143–152. ACM (2010)
9. Kawasaki, Y., Igarashi, T.: Regional Undo for Spreadsheets. In: Proc. UIST 2004 Demostration Abstract. ACM (2004)
10. Kelly, S.U., Davis, P.J.: Desktop History: Time-based Interaction Summaries to Restore Context and Improve Data Access. In: Proc. INTERACT 2003, pp. 204–211. IOS Press (2003)
11. Kobayashi, M., Igarashi, T.: Boomerang: Suspendable Drag-and-Drop Interactions Based on a Throw-and-Catch Metaphor. In: Proc. UIST 2007, pp. 187–190. ACM (2007)
12. Kurlander, D., Feiner, S.: A Visual Language for Browsing, Undoing, and Redoing Graphical Interface Commands. In: Visual Languages and Visual Programming, pp. 257–275. Plenum Press (1990)
13. Meng, C., Yasue, M., Imamiya, A., Mao, X.: Visualizing Histories for Selective Undo and Redo. In: Proc. APCHI 1998, pp. 459–464. IEEE (1998)
14. Myers, B.A., Mcdaniel, R.G., Miller, R.C., Ferrency, A.S., Faulring, A., Kyle, B.D., Mickish, A., Klimovitski, A., Doane, P.: The Amulet Environment: New Models for Effective User Interface Software Development. IEEE Transactions on Software Engineering 23(6), 347–365 (1997)
15. Nakamura, T., Igarashi, T.: An Application-Independent System for Visualizing User Operation History. In: Proc. UIST 2008, pp. 23–32. ACM (2008)
16. Rekimoto, J.: Time-Machine Computing: a Time-centric Approach for the Information Environment. In: Proc. UIST 1999, pp. 45–54. ACM (1999)
17. Seifried, T., Rendl, C., Haller, M., Scott, S.: Regional Undo/Redo Techniques for Large Interactive Surfaces. In: Proc. CHI 2012, pp. 2855–2864. ACM (2012)
18. Shao, B., Li, D., Gu, N.: An Algorithm for Selective Undo of Any Operation in Collaborative Applications. In: Proc. GROUP 2010, pp. 131–140. ACM (2010)
19. Vratislav, J.: Cascading undo control. In: Bachelor Thesis, pp. 1–52. Czech Tecnical University, Prague Faculty of Electrical Engineering (2008)
20. Yeh, T., Chang, T.H., Miller, R.C.: Sikuli: Using GUI Screenshots for Search and Automation. In: Proc. UIST 2009, pp. 183–192. ACM (2009)