Data-driven Adaptive History for Image Editing

Hsiang-Ting Chen* Li-Yi Wei† Björn Hartmann‡ Maneesh Agrawala‡
*National Tsing Hua University †University of Hong Kong ‡University of California, Berkeley

ABSTRACT

Digital image editing is usually an iterative process; users repetitively perform short sequences of operations, undo them and then redo them using history navigation tools. In our collected data, these undo, redo and navigation constitute about 9 percent of the total commands and consume a significant amount of user time. Unfortunately, such activities also tend to be tedious and frustrating, especially for complex projects. We address this critical issue by *adaptive history*, an UI mechanism that groups relevant operations together to reduce user workloads. Such grouping can happen at various history granularities. We present two that have been found to be most useful: On a detailed level, we group repeating commands patterns together to facilitates the *smart undo* functions; On a coarser level, we segments commands history into chunks with similar semantic meaning for easier *semantic navigation*. The main advantages of our approach are that it is easy for users to learn and easy for developers to integrate into any existing tools with text-based history lists. Unlike prior methods that are predominately rule based, our approach is data driven, and thus adapts better to common editing tasks which exhibit sufficient diversity and complexity that may defy predetermined rules or procedures. We evaluate our system via both quantitative analysis and qualitative studies.

Keywords: image editing, interaction, adaptive history, smart undo, semantic navigation

Figure 1: Data-driven adaptive history. Our system (left) provides an adaptive history with better appearance and interaction than the traditional history list as in Photoshop (right). Our prototype system is implemented as a standalone companion application (running on a table) for Photoshop (running on a PC), so that they display synchronized image editing state through Photoshop Connection SDK. All our visualizations (including color coding and command grouping) as well as interactions are achieved in real-time. Please refer to our video for more demonstrations.

INTRODUCTION

Undo, redo, and history navigation are the most commonly used operations in interactive image editing [17]. In our data collected from digital artists, such meta commands constitute about 9% of all issued commands (66 out of 797 per task average). However, they are also the most tedious, as it may take many repetitive operations to reach the intended points on the editing histories. This can greatly diminish user efficiency and satisfaction in common image editing tasks.

Prior works attempted to address this critical issue by proposing new graphical history representations or better undo mechanisms. However, these methods either introduce completely different representations from the traditional linear histories and thus present usability issues (e.g. hierarchical graphs in Chen et al. [4]) or use hardcoded rules that might not be flexible enough in handling more complex real-world cases (e.g. regular expressions in [5] and spatial filter in [20]). Thus, a linear text list, despite its deficiencies, remains the dominant form of operation history representation in image editing softwares .

We propose *adaptive history*, a data-driven method to aggregate user editing operations for more efficient undo, redo, and history navigation (Figure 1). Our two key observations are that 1) users often mentally chunk sequences of low-level editing operations into some higher-level, semantically meaningful units; during undo operations, users typically seek to undo an entire semantic unit instead of just individual operations, and 2) there are certain repeating command patterns with frequent occurrences in users command histories which are semantically inseparable (e.g. copy + paste or patch tool + patch selection in Photoshop). The challenge is to identify the boundaries between such semantic units. By collecting and analyzing actual data from professional artists through instrumented image editing tools, we use machine learning algorithms to train a classifier that can in turn predict the boundaries between semantic units. We employ two particular machine learning methods for two levels of editing history granularity:

Semantic navigation On a coarse level, we use support vector machines (SVM) to identify high level semantic segments, such as skin smoothing or hair coloring for portrait retouching.

Smart undo On a fine level, we use n-gram analysis to identify operations often grouped together to accomplish specific micro tasks, such as “patch tool, patch selection, deselect” for blemish removal in Photoshop.

Overall, our approach belongs to the emerging trend of data-driven user interfaces (e.g. [7, 24, 9, 18, 16]), with specific applications in interactive image editing and history navigation.

Our approach is easy for users to learn and easy for developers to integrate into existing systems. On the user side, we incur only minimal changes to the appearances and behaviors of traditional linear history lists: for appearances, we enhance traditional linear history lists with visual cues with colors and foldable items; for behaviors, users still interact with our adaptive history lists as usual, but have the option to navigate through semantic chunks rather than just individual commands. On the developer side, all they need to do is to plug in properly trained classifiers into existing system plumbing with corresponding UI changes as described above.

We evaluate our system through user studies containing both quantitative measurements and qualitative feedbacks. Both evaluations are very positive and in particular, our study subjects, all of them professional artists, unanimously commented that our system could be very helpful to their daily work flows.

PREVIOUS WORK

Here we review prior art most relevant to our work.

Operation Sequence Analysis

Data analysis has been extensively applied for operation sequences of content creation tools. However, efforts so far have been primarily focusing on text editing, such as word processing [11, 19] and software development [21]. Similar efforts for image editing have started to emerge only very recently [17, 16, 18], probably due to the difficulty of instrumentation. These works have demonstrated the versatility of applying data analysis for image editing, such as command vocabularies [17], command recommendation [18], and UI customization [16]. Such data driven analysis has yet to be applied for facilitating common undo, redo, and history navigation operations, which our work targets.

Graphical History

Graphical history visualization has long been an active research field; See Heer et al. [10] for a comprehensive survey. Here, we focus on works most relevant to operation history in real-world settings (where each history may contain hundreds of commands). These focus on two main strategies: clustering and filtering.

The basic idea behinds *clustering* is to group similar or related commands together to reduce visual clutter [14, 22, 7]. Some recent works include Chronicle from Grossman et al. [8] which builds the history hierarchy based on the user "save" commands, Meshflow from Denning et al. [5] which clusters operations based on hand-crafted regular expression rules, and Chen et al. [4] which visualizes editing histories via hierarchical graphs.

The basic idea behinds *filtering* is to selectively display information based on contextual criteria, including spatial relationships [20, 22, 5, 4], temporal relationships [8, 5], and content properties [13, 2, 23].

A notable difference between these prior techniques and our method is that they are all heuristic or rule-based whereas ours is data driven. Such hand crafted rules or heuristics are usually difficult to derive and have built-in assumptions that can break when facing the diversity of command history in real-world cases. A data-driven approach could instead learn from such diversity and provide better results.

Undo Management

Navigating editing history is one of the most common operations for interactive editing tools, in particular undo and redo. For digital content creation, Meng et al. [20], Su [23] and Chen et al. [4] provided the functionality of "undo by selection", and Edward et al. [6] allowed users to bound atomic operations into semantic "chunks" for better navigation while also prevent accidental undo that breaks such chunk. Inspired by Edward et al. [6], the short-term undo function in our system also serves a similar purpose: "smart undo" for semantic groups instead of just individual operations. However, instead of manually defined rules as in these prior works, we learn the operation patterns from analyzing collected user data. Thus, our algorithm is not constrained to any predefined rules and thus can adapt to different users, tasks, and tool versions.

USER INTERFACE

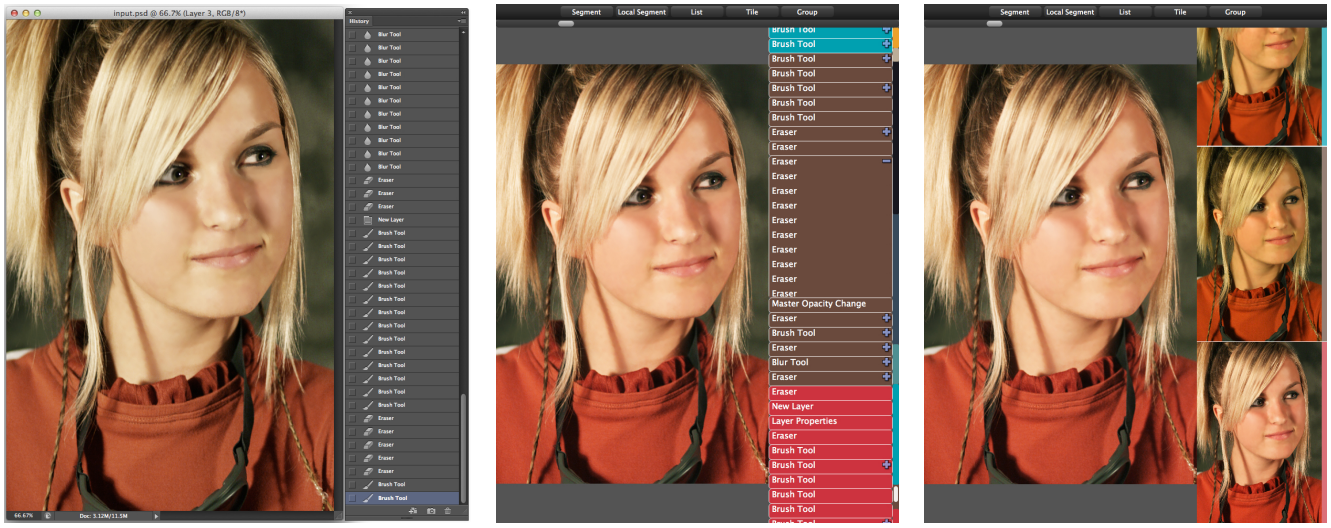
The history panel serves as a core UI component in common image editing systems such as Photoshop and GIMP, with main usages including undo, redo and history navigation. However, its basic representation and interaction have not evolved much from the original linear text-based lists. We propose a data-driven adaptive history with two key enhancements over traditional history lists: semantic navigation at a coarse level and smart undo at a fine level.

This two-level design was based on our informal discussion with artists as well as field observations on their workflows. In particular, we noticed that artists tend to work on two different levels of granularity:

- On a coarse level, they usually used layers or files to manage and store progress milestones, e.g. finished eye retouching or skin smoothing.
- On a fine level, they frequently undid through history list progresses that were considered to be unsatisfactory or exploratory.

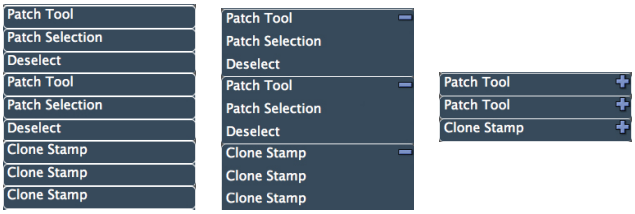
Our *semantic navigation* and *smart undo* have been designed in response to these two levels of common artist workflows. Even though prior work has proposed fancier multi-resolution navigation (e.g. [4, 5]), we believe this is not necessary, due to both prior user studies (e.g. "users tended to stay on a single resolution to avoid losing context of their navigations" as in [4]) as well as our own observations as described above.

From the user's perspective, our interface design minimally changes the feel and look of the traditional history lists. In particular, users still interact with our adaptive history as usual, but with enhanced interaction through semantic navigation and smart undo, and enhanced perception through color coding and foldable tile-based command grouping. From the developer's perspective our interface design, due to its minimal change from traditional history lists, should be straight-



(a) Photoshop history list (b) Our history with semantic segments (c) Our semantic navigation

Figure 2: (a) is the Photoshop history list. (b) is our data-driven adaptive history list. Our UI assigns a unique color to all entries belonging to the same semantic segment, with the scroll bar (right) colored accordingly. Our UI also groups repeating commands into tiles, which can be folded/unfolded via the corresponding plus/minus signs. Double clicking a thumbnail image would lead users to the corresponding segment in the history list. (Note: all screen shots display the same image; the perceived color differences are caused by different color spaces of Photoshop and our Qt-based UI.)



(a) List (b) Tile (c) Folded tile

Figure 3: History list display modes. Our history list has three display modes: list, tile, and group. We provide zoom-in views from our main UI in Figure 2 for easier comparison. List mode (a) is the basic text-based list with semantic colors, tile mode (b) groups the detected repeating commands into a single tile, and group mode (c) folds the command groups in (b) into single entries. Users can also manually fold/unfold by clicking on the plus/minus sign on each group tile.

forward to port into their existing UIs. Our underlying algorithms are all data-driven; with proper training data, developers can trained their own classifier as either a fixed preprocess or as an ongoing process that dynamically adapts to user behaviors.

Semantic Navigation

Scenario Alex is in the middle of a photo retouching session. He has just finished up retouching the hair part of the portrait and was performing the skin smoothing. Feeling uncomfortable about the overall feeling of skin, he decided to undo all the skin smoothing work he has done and starts over. At this point, as the history list has already grown too long and the image already contains too many layers, Alex found that he had to go through lots of trial-and-error to figure out to the right roll-back spot via the Photoshop history list (Figure 2a).

With our system, Alex can clearly see the automatically colored and aggregated segments of his work (Figure 2b) and could easily found the right spot of interest. He can also switch to the thumbnail view (Figure 2c) and performs analogous operations.

Design We design our system with simplicity and minimum changes to traditional linear histories. The segments are visualized with different colors and correspond to semantic chunks such as hair retouch, skin smoothing, and eye sharpening. We pick the color scheme with following goals in mind: 1) segment colors should differ sufficiently from each other for distinctiveness, and 2) the colors should be dark enough to visualize the text labels, which we designed to be all white instead of in different colors to avoid confusion. In our early design phase, we have tried to use colors based on the underlying color changes of the corresponding image regions. However, this might not produce distinctive enough colors. For example, the representative colors for retouching eyes and retouching hair can be very similar.

The segmentation is performed on the fly, based on the classifier trained from the data collected from the professional artists. Users can adjust the granularity of the segmentation using the slider bar at the top of the UI (Figure 2b). In addition, we also modify the color of the scroll bar beside the history list, so that users can see the relative lengths of the segments (Figure 2b).

Smart Undo

Scenario During skin smoothing, Alex use the healing patch tool to remove larger area of undesired features on the skin.

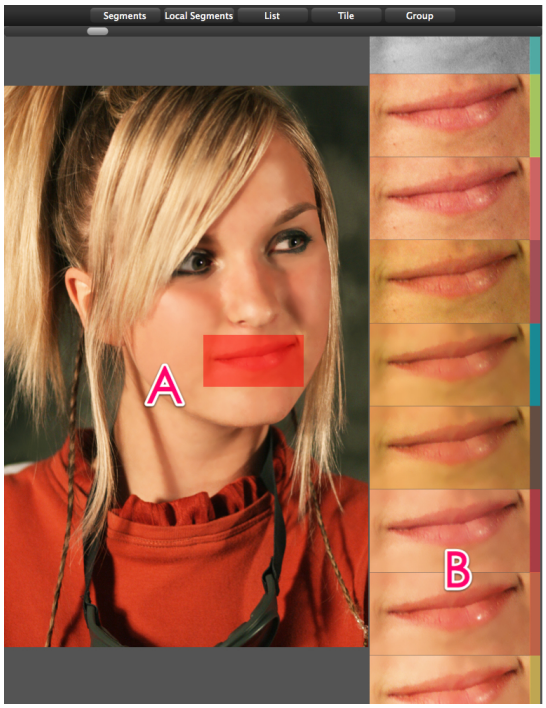


Figure 4: Spatial filter. Users can select particular regions of interest (A), and review the corresponding history via the thumbnail images at (B).

Each removal operation involves three steps: (1) select the target region to be retouched, (2) drag the target region to the artifact-free source region, and (3) deselect the target region. If he is not satisfied with the result, Alex would have to issue the undo commands three times in a row to fully undo the retouching effect. Similar scenarios also happen on other tools such as the clone stamp tool, the pen tool (add anchor point, drag anchor point) and common action patterns such as copy-paste-move. Also as Alex is using the brush tool, the history list is quickly filled with identical "brush tool" entries. If he is not satisfied with the outcome, using the traditional interface he would have to continuously issue undo commands and visually check the result to find the point he wishes to return to.

With our system (Figure 3), such repeating commands are automatically grouping into a foldable single entry. Alex can smart undo the whole chunk of command and navigate through the history list more easily.

Design We provide three different display modes for visualizing the repeating command patterns (Figure 3). Besides the basic command list, users can switch to "tile" mode, where repeating patterns are grouped into single tiles. In "group" mode, tiles are automatically folded into single entry, while users can manually fold/unfold the entry by clicking on the plus/minus icons. In these modes, the smart undo function can undo a whole tile/group rather than a single command entry.

Spatial Filter

Besides two major functions mentioned above, we also provide a spatial filter function to enhance the usability of our system. Spatial filter allows users to select particular image regions of interest and see only the relevant portions of the command history. For example, Alex might want to roll back to the point before he retouched the eyes. With this function, he can select the region of interest and our system will show the thumbnail image of the region in each semantic segmentation (Figure 4).

DATA COLLECTION

In this section, we describe how we collected data to feed into our machine learning algorithms for semantic segmentation (SVM) and smart undo (n-gram).

Instrumentation We picked Photoshop as our target software due to its popularity among digital artists. The instrumentation is achieved by the *history log tool* provided in Photoshop, which outputs command and parameters in human-readable text form, as well as a Photoshop scripts written by ourselves that recorded intermediate frames of every command in the history. We need such script for recording the complete history because the *history log tool* did not output the stroke motions and parameters of brush-operations.

Artists To collect the training data, we hired 10 freelancer artists on oDesk. They were all professional photo retouchers with at least 5 years of working experiences residing in different countries including Belarus, Bulgaria, Colombia, Philippines, Serbia, Ukraine and United States. The average hourly payment was 14.3 U.S. dollars ($\sigma = 5.4$). Six of them used keyboards and mice as their primary interaction devices while the other four used Wacom pen tablets.

Tasks We then gave each artist four portrait photos downloaded from Flickr for retouching. We were aware of the fact that some artists may prefer to perform minimal retouching to keep the portraits as natural as possible. To avoid such situation and make sure our collected data contains enough information, we gave artists a minimum retouch requirement list, e.g. you mush retouch the hair, whiten the eyes, remove blemish, etc. Interestingly, in the end we noticed that all retouchers actually performed much more retouches than we asked. We believed this implied that these professional retouchers truly respected their works and had a very high bar on the quality of the portrait shot. For the retouching of all four photos, the average working time was 9 hours and 28 min ($\sigma = 1$ hour and 55 min) and the average number of collected operations per artist was 852 ($\sigma = 444$, $min = 893$, $max = 5977$).

After done retouching, we asked the artists to segment and label their own retouching processes into smaller sub-tasks based on the reference retouching tag lists we provided them. The list was compiled with the chapter/section names of the photo retouching tutorial book [12] with entries like "eye sharpening", "blemish removal", "skin smooth", etc (complte list at Appendix). We also made it clear to artists that they can add their own tag descriptions if necessary.

In summary, we collected three types of data: fine-grained operation sequences (in text), corresponding image content of each operation, and the text labels describing the operation sequences provided by the artists.

ALGORITHM

We analyze the collected data with machine learning approaches; SVM (Support Vector Machine) classifier for semantic segmentation for semantic navigation and n-gram analysis for extracting repeating patterns for smart undo.

The semantic segments typically consist of tens to hundreds of commands thus we deploy SVM classification to facilitate its particular strength in handling high-level feature vectors. While the repeating patterns are often much shorter and thus we extract it via a light-weighted n-gram approach that basically only involves efficient hash table searching and substring matching.

Semantic Segmentation

Observation Some meta operations, such as layer creation, have a higher probability than others to correspond to semantic segmentation boundaries. However, simply treating every layer creation command a as segmentation boundary may not produce the desired outcomes. For example, skin smoothing often involves the creation and combination of multiple layers, which should belong to one instead of multiple semantic segments.

Our key observations are 1) some commands are more likely to appear between the semantic segments, e.g. layer creation, layer merge, and 2) operation sequences spanning multiple segments usually contain very different command sets and modify different image regions (corresponding to different segments), if compared to those belonging to a single segment. The implication is that a properly trained classifier based on both operation histograms and image contents should perform better than methods based on operation types alone.

Operation Space Dimensionality Reduction We trained our classifier in the operation space. However, there are about 168 basic operations in Photoshop (command information obtained from Photoshop → Edit → Keyboard Shortcuts → Summarize). The large number of operations can hamper the ability of our analyzers to extract meaningful information from collected user editing sequences.

To address this issue, we first reduce the dimensionality of operation space before applying our analysis algorithm. Among these operations, some have similar semantic meanings, such as ("New Layer", "New Color Filled Layer", "Layer from Background Color") and ("Rotate 90 degree clockwise", "Rotate 180 degree clockwise"), while others are designed for similar purposes, such as 15 different blurs in Photoshop. Fortunately, the Photoshop menu hierarchy already provides good suggestions about command clusters; for example, all blurs commands are clustered under the menu item of "blur". This allows us to reduce the dimension of the command space from 168 to 41 by simply classifying all non-root-level commands into its ancestor command at the root level menu.

Classification We treat the problem of finding segmentation points as a binary classification: given an input operation sequence, the trained classifier labels each operation either as "segmentation point" or not based on the predicted floating-point probability.

We associate a feature vector with each operation; the vector is defined as the combination of the operation histogram and corresponding affected image regions of the operations preceding the associated one. (Note that we use a causal window so that we can perform on-the-fly prediction based on users' current operation.) Formally, for the j_{th} operation, we define its previous k operations as its neighbor $N(j)$ and its feature vector $F(j)$ can be written as:

$$F = \{O, P\} \tag{1}$$

where:

- $O \in R^o$ indicates the histogram of operations in N, R^o the o-dimensional (reduced) operation space, and $O[i]$ the number of the operations of type i appears in N.
- $P \in R^p$ is the position vector; we divide the image into p grids (default $p = 16$), and P_i equals to 1 if the content at i_{th} grid is modified and 0 otherwise.

With the feature vector defined as above, we train the SVM classifier using LIBSVM [3].

Validation To exam the precision of our SVM classifier, we performed a 4-fold validation. In our data set, each of 10 artists was asked to retouch and label four photos. We trained the classifier from 3 of the 4 photos (i.e. 30 photo retouch history in sum) and predict the one left (i.e. 10 photo retouch history). The average prediction precision is 96%.

However, our data set is asymmetric where most positions are negative and only very few of them, which lie on segmentation boundaries, are positive. If we only look at the positive label and ignore the negative ones, the prediction precision is 70%. We believe that the lower prediction rate is due to the different granularity of labeling in our data set. For example, some artists might label the whole chunk of operations as *face retouch* while others might separate it into *skin smoothing*, *blemish removal*, etc. As a result, applying a classifier with finer granularity to data with coarse labels will produce many false positive labels. Fortunately, in our user study, such false positive results did not turn out to negatively affect users' experience of semantic navigation, as it is still better than traditional non-segmented history lists.

Smart Undo

Observation There are many short repeating patterns with high frequency and are semantically inseparable. Common examples include "Patch Tool, Patch Selection, Deselect" for skin retouch, "Add Anchor Point, Drag Anchor Point" for path selection, and "Master Opacity Change, Merge Layer" for finishing retouching sub-tasks. The inseparability means that it is usually meaningless to undo or navigate to the middle of such chunk. This leads to our idea of highlighting the last location of these small chunks for easier navigation and the smart undo for entire chunks instead of individual commands.

N-gram Analysis To extract the useful repetitions, we perform standard n-gram analysis ($n >= 3$). If the patterns occur more than three times , we insert it into a hash table. There are lots of redundant patterns in the n-gram table, e.g. the pattern of $ABCABC$ is the complete repetition of ABC; and we should only insert the ABC pattern

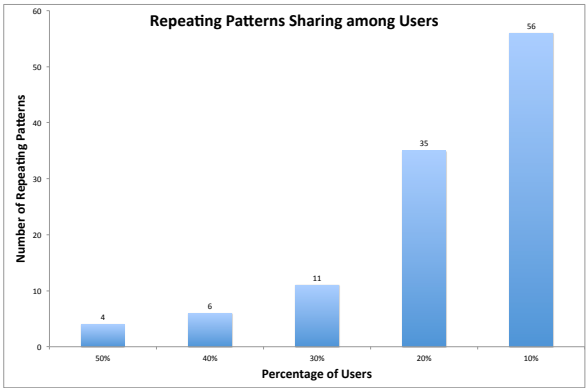


Figure 5: Pattern usage chart. There are 4 patterns (average length 3) shared by half of the artists while 56 patterns (average length 3.7) used by only one artist.

Pattern	Count	User%
Eraser, Master Opacity Change, Eraser	26	50%
Move, Free Transform, Move	11	50%
Master Opacity Change, Hue/Saturation Layer, Modify Hue/Saturation Layer	8	50%
Brush Tool, New Layer, Brush Tool	7	50%
Patch Tool, Patch Selection, Deselect	295	40%
Blending Change, Master Opacity Change, Blending Change	12	40%

Table 1: Pattern usage table. Here we shows top patterns that are used by most artists.

Pattern	Count	User%
Healing Brush, Layer Via Copy, Gaussian Blur, High Pass, Curves Layer, Modify Curves Layer, Group Layers, Blending Change, Channel Mixer Layer, Modify Channel Mixer Layer	4	10%
Duplicate Layer, High Pass, Gaussian Blur, Invert, Blending Change, Blending Options, Add Layer Mask, Invert, Brush Tool	4	10%
Duplicate Layer, High Pass, Blending Change, Add Layer Mask, Invert, Eraser	5	10%
Master Opacity Change, Duplicate Layer, Merge Layers, Selective Color Layer, Fill, Brush Tool	4	20%
Lasso, Paste, Layer Order, Move, Free Transform, Eraser	5	10%

Table 2: Table with longer command patterns. There are some longer repeating patterns, but they are usually the reflection of users' personal habit or only shared by small amount of users.

into our pattern hash table. We solve such redundancy by a top-down $\rightarrow$ bottom-up approach. We first scan the table from top-down (larger n to smaller n), and remove the patterns that are complete repetitions of some existing patterns (e.g., $ABCABC$ is the complete repetition of ABC). Then in second pass, we perform a bottom-up scan that removes patterns which are sub-strings of other longer patterns (e.g. ABC is the sub-string of $ABCD$).

Repeating Pattern By performing the n-gram analysis, we extract command patterns with length equal to or longer than 3 and occur more than 3 times in the data. We found 112 such patterns with the average occurrence count of 7 (σ= 27).

Figure 5 shows that users usually have their own particular personal usage patterns but about half of them are still shared by two or more users. While Table 1 and Table 2 show that shorter command patterns are more likely to be shared among users while longer ones usually reflects the personal usage patterns.

The command pattern with high number of occurrence across users imply that it should be treated as an atomic operation in the history list and thus lead to our idea of "smart undo". Such pattern can also serve as precious data to improve the design of image editing softwares or as macros to distribute over the community [1, 16].

Identical Repetition It is common for users to issue identical commands repeatedly, such as brush, clone tool or move. Some previous works counts all such repeating commands as single entries [15, 16], which we believe is not correct since users actually spent most of their time on such repetitive operations and actions of same command type could still have different semantic meanings (e.g. brush operations for retouching hair and retouching skin should not be clustered together). In our system, we group such identical repeating commands based on the position of modified region and time spent on operations. More specifically, we group commands together if they modify the same region of the image (based on the position vector) or is issued in less than 500 ms interval. (Note that parameters of operations could also serve as a good factor for grouping, but currently it is not possible to obtain parameters related to brush operations in Photoshop due to the SDK constraint.)

SYSTEM IMPLEMENTATION

We built our system as a standalone application instead of part of Photoshop; the core algorithms are implemented with Qt and the user interface with QML. With newly announced Connection SDK from Adobe, our application communicates and synchronizes with Photoshop via TCP protocol.

It is certainly possible to implement our system via Photoshop plug-in for direct integration, allowing users to perform all the interaction and editing in one place. However, after some considerations, we chose not to do so for our first prototype because the plugin SDK imposes too many limitations on UI design. For example, it is not clear how we can create important UI components in our system such as a list with foldable items or a scrollbar with multiple colors through Photoshop plug-in SDK. Another main reason for using Connection SDK is that our standalone application can run on a variety of other devices such as tablets and smartphones to provide additional flexibility and screen space.

One implementation detail worth mentioning is that due to the latency and performance issues, it is currently not possible for Photoshop to transmit every intermediate frame to clients, especially when users issue commands in a high frequency or when the images are large. Our solution is to synchronize only the text history list when user is actively interacting with Photoshop, and sync the intermediate images in undo stack only when Photoshop is idle.

EVALUATION

We evaluate and compare our system with traditional linear text history lists and time-based clustering of command history in [18] through both quantitative user studies and qualitative user feedbacks.

We recruited 6 professional artists who used Photoshop on daily basis in their works, including two professional photo retouchers, three professional photographers, and one digital image sketcher. The test session runs about one hour in average and we paid each artist 30 dollars for the studies.

Quantitative Study

Scenario We constructed the user study scenario as follows. The user was in the middle of the photo retouching process; dissatisfied by the overall progress, she would like to roll-back to some previous states. We pre-recorded three photo retouching processes from professional artists and annotated the videos with high-level semantic descriptions, such as *skin smoothing*, *eye retouching*. We asked testers to follow such prerecorded videos instead of giving them completely freedom to improvise for two main reasons. First, without some predefined rules or constraints, the testers might not use the functions of interest enough (e.g. undo and history navigation) and they might not even use history list at all (e.g. some testers were used to manage progress via layers). Second, our user study involves the comparisons between three different systems and a pre-recorded scenario could provide a better control.

Session Each test session started with a 5 minute introduction to our system and its features, followed by asking the testers to perform several simple trial tasks such as "fold/unfold the command group", "perform partial comparison on eye region", and "increase the number of semantic segments via slider".

After the simple trials, we briefly described the comparison study we were going to conduct as well as the two other systems involved, i.e. the text-based history and the time-based one. Then we conducted the official comparison studies. Specifically, we asked the testers to perform the following three tasks on all three photos using three different systems in turn (e.g. first photo with text-based history system, second photo with time-based system and third photo with ours):

1. Located the approximate start point of eye retouching
2. Located the approximate start point of skin smoothing
3. Located the approximate start point of lip retouching

Given the nine different combinations between three photo sets and three systems and six different permutations on the order of using these three systems, we have carefully counterbalanced these factors by adjusting the order and photo/system pair during the test session as well as randomizing the order of these three tasks. We measured the quantitative data based on rate of success, time to completion, and number of mouse clicks over each session.

Result Table 6 showed the result of the comparison test. We also performed the one-way ANOVA on three measured

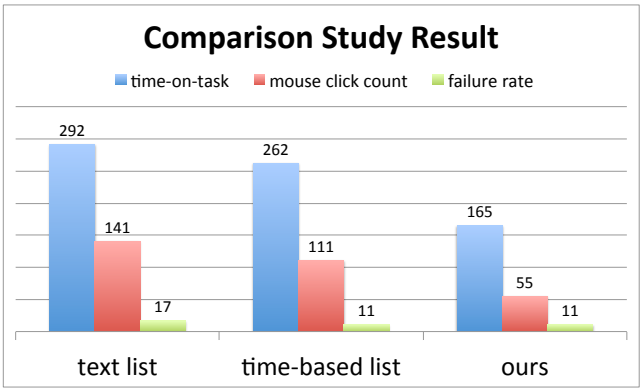


Figure 6: Comparison study chart. We use failure-rate instead of success-rate (as in the main text) for more coherent visualization, so that all quantities are the lower the better.

factors: time-on-task: $F = 11.83, p\text{-}level = 0.0014$; mouse click count: $F = 10.56, p\text{-}level = 0.0022$; success rate: $F = 0.75, p\text{-}level = 0.49$. The lack of sufficient differences on the success rate indicates that testers were pretty good at finding the target states using all three kinds of systems. But for two other measurements, time-on-task and mouse-click-count, the differences are significant and our system exhibited significant improvements over the other two systems.

Qualitative Feedback

After each quantitative study session, we gave each tester additional 20 minutes to explore our system and soliciting their opinions. Our 7-point Likert survey consisted of questions regarding two major aspects: 1) test case scenarios and 2) our system features. The first part was to make sure the designed scenario was sensible while second part was for evaluating our system design.

On Test Case Scenario On the design rationality of the test case scenarios, we asked two questions. On the first question "your level of understanding on the recorded photo retouch process", the average score was 6.3 (two testers with lowest score at 5). When we were designing the test cases, one of the biggest concern was that testers might not be able to comprehend others' retouching process. However, during the test session, most testers actually mentioned that they could easily understand the retouch techniques upon watching the retouched image and did not mind us fast-forwarding or even skipping the video as long as we provided them a simple list of retouching order. It showed that these professional artists shared their own professional languages on photo retouching and relieved our initial worry on the design of our test cases. On the second question "does our designed scenario usually happen in your daily retouching work?", the average score was 6 (three testers with lowest score at 5). They all agreed that the scenario of rolling back to some previous states happened a lot during their daily works. The three low scores of 5 came from the comment that there are other alternative approaches for rolling back than history list navigation, e.g. layers or the history snapshots functions.

On System Features On features of our system, the responses from testers were very positive (Figure 7) and they

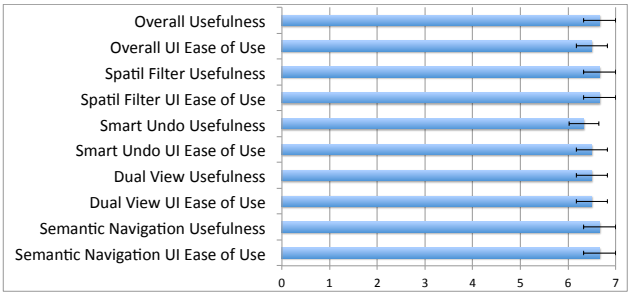


Figure 7: Likert scale rating for importnat functions in our system. Error bars show 95% CI.

all genuinely looked forward to see these new features being added over the current dull history list of Photoshop. Averaged across all features, the ranking were 6.5 for both "ease of use" and "usefulness". One tester gave lower scores on semantic navigation (5) and dual view (4) and he commented that he has to deal with hundreds of photos in limited time and thus the history list for his retouching tends to be quite short and he does not think the semantic segmentation would be very useful for him. On the other hand, he gave the command grouping function the score of 7 for he believed such function could make the history list more compact and lead to better efficiency.

We also received many encouraging comments from our testers. One professional photographer who seldom uses layers but mostly use history list and snapshot for daily photo retouching works said "The history list hasn't changed too much since around Photoshop 5.0, which added the function of snapshot; your system would be really really useful for my workflow and I cannot wait to use it". The image sketcher said "I appreciate the fact that you add these cool features without increasing the size of history list. History list review is very important to me, and although there are plug-ins available, they often took additional spaces and distracted my work". And from the professional retoucher "I mainly manage my progress via layers because the history list in Photoshop is really not that helpful. With your system, it seems like that I could finally use fewer layers in my workflow!".

LIMITATIONS AND FUTURE WORK

Data Variety We have designed our UI scenarios and system algorithms for general digital image editing, even though only portrait retouching is considered for the data collection and evaluation parts of our project. This is mainly due to the need for project cost and budget control. We chose photo retouching as the primary subject of our pilot study for the following reasons: it is one of the most common and popular type of photo editing tasks among professional photo retouchers, and usually contains sufficiently long, varying, and complex editing histories to benefit from good navigation mechanisms as well as contribute to data analysis + machine learning. As a future work, we would like to collect data and perform studies from different kinds of image editing tasks, such as scenery retouching, graphics design, and digital sketching.

Adaptive Learning Our current implementation performs both SVM learning and n-gram analysis as an offline preprocess based on the collected data from professional artists. This fixed initialization might not adapt well to individual users or tasks with different preferences or characteristics from the initial training data. One interesting potential is to perform both learning processes periodically in an incremental fashion so that our system can better adapt to individual users and tasks. However, doing so requires cares in both the UI design and system implementation aspects. In the UI part, past user studies indicated that an adaptable UI may confuse users in certain situations [16]. In the system implementation part, the n-gram analysis is already real-time as it involves only hash-table look-up. But the SVM classifier would take minutes for training, which is not fast enough for online learning. We are looking at potential incremental learning algorithms as well as the possibility of stealing idle CPU times for such training tasks.

Nonlinear navigation Our system currently supports only linear undo, redo, and navigation. As demonstrated in earlier works such as Chen et al. [4], nonlinear explorations can provide extra flexibilities and possibilities not possible with linear exploration, such as undo a specific image region (e.g. left eye of a portrait) with histories preceding other regions that we wish to keep (e.g. right eye of the same portrait). We believe extending our current methodology from linear lists to non-linear graphs should be quite feasible from the algorithm and design perspective. However, the main issues lie in usability; non-linear histories will differ more from traditional linear histories than our linear adaptive approach. Thus, further user studies will be needed for this.

Segment Labeling We are also interested in the possibility of automatic labeling the semantics segments extracted from command histories. In our preliminary experiments, we have found it a difficult task due to the high amount of ambiguity among different photo retouching techniques. For example, artists might use very similar operation sets for different tasks (e.g. retouching eyes and hairs), and they might also use very different operation sets for similar tasks (e.g. skin smoothing). We believe that with better similarity measurements (e.g. region segmentation for facial structures in Berthouzoz et al. [1]), it should be possible to automatically label the command histories.

CONCLUSIONS

In this paper we introduced a *data-driven adaptive history* that enhances the usability of traditional linear history with two main mechanisms: *semantic navigation* and *smart undo*. Our design minimally changes the look and feel of traditional history lists, while significantly enhances their usability and satisfaction for history navigation. Users can easily adapt to our UI, while the developers will find it straightforward to integrate our design into common image editing systems. Our methodology is data-driven; with proper data collection, it can be applied to different image editing tasks and systems. Our design has shown significant improvements over traditional history lists through user studies that incorporate both quantitative measurements and qualitative feedbacks. We believe that our data-driven history list could be of great benefit

to digital artists, who tend to have long editing histories due to the complexity and the trial-and-error nature of their common tasks.

REFERENCES

1. F. Berthouzoz, W. Li, M. Dontcheva, and M. Agrawala. A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Trans. Graph.*, 30:120:1–14, 2011.
2. S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: visualization meets data management. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747, 2006.
3. C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. T. Chen, L.-Y. Wei, and C.-F. Chang. Nonlinear revision control for images. In *SIGGRAPH '11*, pages 105:1–10, 2011.
5. J. D. Denning, W. B. Kerr, and F. Pellacini. Meshflow: Interactive visualization of mesh construction sequences. In *SIGGRAPH '11*, 2011.
6. W. K. Edwards, T. Igarashi, A. LaMarca, and E. D. Mynatt. A temporal model for multi-level undo and redo. In *UIST '00*, pages 31–40, 2000.
7. F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi. Generating photo manipulation tutorials by demonstration. In *SIGGRAPH '09*, pages 66:1–9, 2009.
8. T. Grossman, J. Matejka, and G. Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *UIST '10*, pages 143–152, 2010.
9. B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer. What would other programmers do: suggesting solutions to error messages. pages 1019–1028, 2010.
10. J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1189–1196, 2008.
11. J. Kay and R. C. Thomas. Studying long-term system use. *Commun. ACM*, 38(7):61–69, July 1995.
12. S. Kelby. *Professional Portrait Retouching Techniques for Photographers Using Photoshop*. Peachpit, 2011.
13. D. Kurlander and E. A. Bier. Graphical search and replace. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 113–120, New York, NY, USA, 1988. ACM.
14. D. Kurlander and S. Feiner. Editable graphical histories: the video. In *CHI '91*, pages 451–452, 1991.
15. D. Kurlander and S. Feiner. A history-based macro by example system. In *UIST '92: Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 99–106, 1992.
16. B. Lafreniere, A. Bunt, M. Lount, F. Krynicki, and M. A. Terry. Adaptablegimp: designing a socially-adaptable interface. In *UIST '11 Posters*, pages 89–90, 2011.
17. B. Lafreniere, A. Bunt, J. S. Whissell, C. L. A. Clarke, and M. Terry. Characterizing large-scale use of a direct manipulation application in the wild. In *Proceedings of Graphics Interface 2010*, GI '10, pages 11–18, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.
18. W. Li, J. Matejka, T. Grossman, J. A. Konstan, and G. Fitzmaurice. Design and evaluation of a command recommendation system for software applications. *ACM Trans. Comput.-Hum. Interact.*, 18:6:1–6:35, July 2011.
19. F. Linton and H.-P. Schaefer. Recommender systems for learning: Building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction*, 10(2-3):181–208, Feb. 2000.
20. C. Meng, M. Yasue, A. Imamiya, and X. Mao. Visualizing histories for selective undo and redo. In *APCHI '98: Proceedings of the Third Asian Pacific Computer and Human Interaction*, page 459, 1998.
21. G. C. Murphy, M. Kersten, and L. Findlater. How are java software developers using the eclipse ide? *IEEE Softw.*, 23(4):76–83, July 2006.
22. T. Nakamura and T. Igarashi. An application-independent system for visualizing user operation history. In *UIST '08*, pages 23–32, 2008.
23. S. L. Su. Visualizing, editing, and inferring structure in 2d graphics. In *Adjunct Proceedings of the 20th ACM Symposium on User Interface Software and Technology*, pages 29–32, 2007.
24. S. L. Su, S. Paris, and F. Durand. Quickselect: history-based selection expansion. In *GI '09: Proceedings of Graphics Interface 2009*, pages 215–221, 2009.

Appendix: Reference Label List

The reference label list provided to hired artists during data collection.

- global adjustment
  - color/tone adjustment
  - brightness/contrast adjustment
  - transformation (crop/rotation/perspective)
- retouching eyes
  - sharpening the eyes
  - brightening the whites of the eyes
  - removing eye veins
  - reducing dark circles under eyes
  - deforming eye
  - enhancing eyelashes
  - enhancing eyebrows
- retouching skin
  - smoothing skin
  - removing hotsopt
  - removing blemishes
  - removing wrinkle
  - applying digital makeup
- retouching hair
  - removing stray hair
  - filling hair gap
  - adding highlight to hair
- retouching lips
  - changing lip shape
  - changing lip color
  - creating glossy lip
  - whitening/repairing teeth
- slimming and trimming
  - face thinning
  - body slimming