Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials

Amy Pavel
Floraine Berthouzoz
Björn Hartmann
Maneesh Agrawala

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-167
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-167.html

October 9, 2013

Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials

Amy Pavel, Floraine Berthouzoz, Björn Hartmann, Maneesh Agrawala - UC Berkeley EECS

ABSTRACT

Many people rely on Web-based tutorials to learn how to use complex software. Yet, it remains difficult for users to systematically explore the set of tutorials available online. We present Sifter, an interface for browsing, comparing and analyzing large collections of image manipulation tutorials based on their command-level structure. Sifter first applies supervised machine learning to identify the commands contained in a collection of 2500 Photoshop tutorials obtained from the Web. It then provides three different views of the tutorial collection based on the extracted command-level structure: (1) A *Faceted Browser View* allows users to organize, sort and filter the collection based on tutorial category, command names or on frequently used command subsequences, (2) a *Tutorial View* summarizes and indexes tutorials by the commands they contain, and (3) an *Alignment View* visualizes the command-level similarities and differences between a subset of tutorials. An informal evaluation (n=9) suggests that Sifter enables users to successfully perform a variety of browsing and analysis tasks that are difficult to complete with standard keyword search. We conclude with a meta-analysis of our Photoshop tutorial collection and present several implications for the design of image manipulation software.

ACM Classification

H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

Author Keywords

Web-based tutorials, photo manipulation, faceted browsing

INTRODUCTION

Image manipulation software offers a suite of sophisticated tools for users to edit their photographs. Yet, these tools are often very complex and therefore difficult to learn and use effectively [14]. As a consequence, users commonly turn to task-centered tutorials to learn how to use the software [28, 4]. These tutorials use narrative text along with interface screenshots to describe the sequence of commands necessary to achieve a specific goal (Figure 1). Sites such as

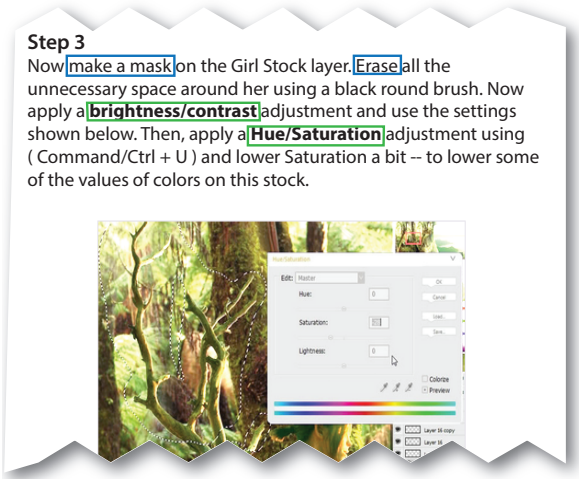

Figure 1. A step from an example online tutorial. Tutorials use direct references to commands (i.e. exact string matches to the command name, shown in green) and colloquial/indirect references to commands (e.g. tutorial says "make a mask" rather than writing the command name "Layer Mask", shown in blue)

`tutorialized.com` or `good-tutorials.com` contain tens of thousands of such tutorials.

The large number of online tutorials and their diverse formats can make it difficult for users to systematically browse and compare tutorials. Current websites simply organize tutorials by high-level categories such as "Photo Effects" or "Web Layouts". To explore the collection of tutorials users must either page through long lists of tutorial titles within each category or perform keyword searches on the natural language tutorial text. None of these websites exploit the underlying command-level structure (i.e., the sequence of software commands) of the tutorials.

Access to such command-level structure could greatly facilitate many search and browsing tasks. Prior work found that tutorial users are interested in identifying both "useful" and "familiar" commands [18]. Commercial software expertise sharing systems are also beginning to emphasize the importance of commands for searching, filtering and reviewing interactive tutorials [1]. Command-level structure can help a novice Photoshop user identify frequently used commands; the novice user may choose to learn them before other, less common commands. Expert users who are familiar with a subset of commands could quickly find tutorials that leverage their existing knowledge in new settings.

Beyond single commands, users may search for subsequences of commands that appear frequently as these subsequences

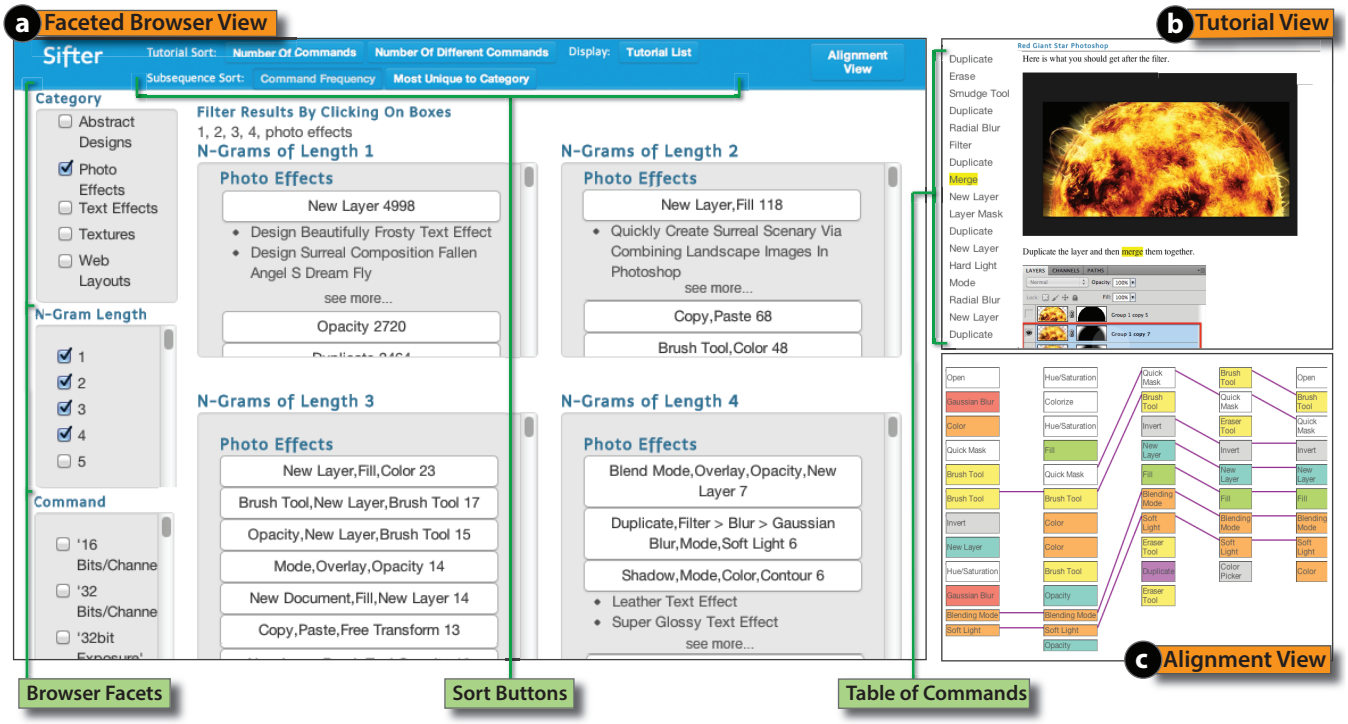

Figure 2. Sifter allows users to explore our collection of 2500 Photoshop tutorials based on their command-level structure. In the *(a) Faceted Browser View* users can organize tutorials using the Category, the N-Gram length (i.e. command sequences of length *N*) and the Command name facets. Clicking on a tutorial title loads the tutorial webpage and its table of commands into the *(b) Tutorial View*. Users can also align the command sequences of multiple tutorials to see the similarities and differences between them in the *(c) Alignment View*.

may represent successful higher-level strategies. For example, many photo adjustment tutorials include the three command subsequence; "Duplicate Layer", "Sharpen", "Paint Opacity Brush". This strategy blends the original image with the sharpened version and allows spatial control over the strength of the effect. Finally, large collections often contain multiple tutorials that use different approaches (e.g., different commands) to achieve the same goal. Users may wish to understand the differences between such alternatives.

Today, such command-based search and comparison is extremely difficult for existing Web-based tutorials because tutorial browsing interfaces do not capture, analyze or expose the command-level structure of tutorials. In contrast, online catalogs in other domains such as e-commerce and libraries frequently rely on structured meta-data to offer faceted browsing and feature-based comparison to successfully navigate and query complex collections. Our research goal is to bring this flexibility to the domain of software tutorials.

In this paper, we present *Sifter*, an interface that allows users to browse, compare and analyze image manipulation tutorials based on their command-level structure. Sifter includes an extraction tool that identifies command sequences in natural language software tutorials. To build this tool we extend the machine learning approach of Fourney et al. [11] to automatically identify direct, indirect and colloquial references to commands (e.g. the tutorial says to "make a mask" instead of the command name "Create Layer Mask", Figure 1). We apply our command extractor to a collection of 2500 Photoshop tutorials we obtained from the Web and achieve an average precision of 90% and recall of 99%.

Sifter also provides an interface for exploring tutorial collections based on three different views of the command-level structure (Figure 2). The *(1) Faceted Browser View* allows users to organize, sort and filter the collection. Filtering facets include tutorial category, the length of command subsequences (N-Gram length) and command names of interest. For instance, a user can quickly find the most common multi-command strategies in the "Photo Effects" category by selecting N-Gram lengths 1 through 4 and the "Photo Effects" facet. When a user selects a tutorial in the faceted browser the tutorial webpage appears in the *(2) Tutorial View* along with a *table of commands* that serves as both a summary of the tutorial and an index into the webpage. The *(3) Alignment View* graphically depicts the similarities and differences in the command structure of a subset of tutorials. Together these views allow users to find relevant tutorials, understand gaps in their knowledge of the application commands and explore the repeated command-level patterns used across a variety of image manipulation tasks.

In an informal evaluation of Sifter we asked nine first-time participants to complete a series of tutorial browsing and comparison tasks using our interface as well as standard keyword search. Qualitative user feedback indicates that the Sifter interface is easy for newcomers to learn and understand and that users find faceted command-level browsing to be useful for exploring large tutorial collections. Moreover, participants successfully completed a variety of novel browsing and exploration tasks with Sifter. We show that these tasks are difficult to accomplish with keyword search. Participants strongly preferred to explore tutorial collections with Sifter

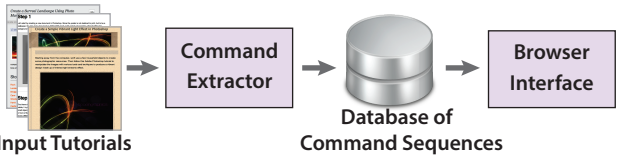

Figure 3. Sifter includes three main components; a command extractor, a database of command sequences and a browser interface.

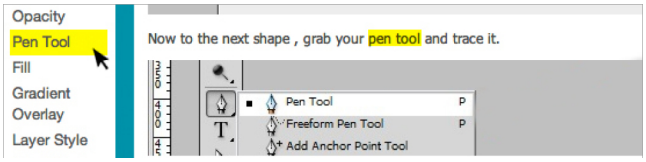

Figure 4. Clicking on the "Pen Tool" command in the table of commands scrolls the tutorial to this command and highlights it in the tutorial.

over keyword search. These results suggest that Sifter is an effective tool for browsing and comparing large collections of image manipulation tutorials.

We conclude with a meta-analysis of our Photoshop tutorial collection to better understand the patterns of command structure both across the collection and within tutorial categories. Based on this analysis present several implications for the design of user interfaces for image manipulation software.

RELATED WORK

Sifter builds on two areas of prior work: techniques for capturing or extracting the command-level structure of software usage and systems that use such data to improve tutorial search and comparison.

Capturing or extracting command-level structure. Researchers have developed techniques for generating tutorials from recorded user actions using instrumented software. This work focuses on generating either static, step-by-step visual tutorials [13], or interactive tutorials [19, 17, 2, 6] or even fully automated macros [3]. Others have used the recordings to create graphical histories that allow nonlinear undo and replay of commands [29, 16, 25, 15, 5, 8]. All of these methods require instrumenting the application source code to record commands. In contrast, our work focuses on developing a lightweight approach for extracting commands from pre-existing text-based tutorials.

Our extraction algorithm is inspired by earlier work on using string matching [27], grammar processing [21] and machine learning techniques [11, 20] to identify text references to commands in software tutorials. Laput et al. [20] achieve the best precision/recall rates using a Conditional Random Field classifier. However, their method requires a relatively large set of 400 training tutorials. Fourney et al. [11] focus on command extraction with sparse training data. They use a Naive Bayes classifier with Witten-Bell smoothing and require only 35 training tutorials while still maintaining good precision/recall. Neither method detects colloquial references to commands. We extend the approach of Fourney et al. to identify colloquial references with sparse training data.

Users also share software tutorials as screencast video recordings. Sikuli [30], Prefab [9] and Pause-and-Play [26] apply computer vision techniques to reconstruct GUI-level interface operations (e.g. menu navigation, button presses, widget clicks, etc.) from such screenshots. However, methods that rely solely on visual GUI output may not be able to robustly identify commands like keyboard shortcuts that do not have visual effects. In contrast, text tutorials usually describe every command that must be executed.

Using command-level structure to improve tutorial search and comparison. Several previous methods use command execution histories to provide more effective application help and tutorial comparison. Ekstrand et al. [10] use the execution history and currently active tools to augment Web search and help users find more relevant tutorials. IP-QAT [23] and LemonAid [7] use recent commands and application context to recommend entries in application-specific question answering systems. CommunityCommands [24] recommends commands for design software based on collaborative filtering algorithms. Unlike Sifter, these tools do not extract information from existing tutorials and they do not enable browsing or analyzing large collections of tutorials.

Closest to our work is Delta [18], an interactive tool that summarizes and compares tutorials based on their command-level structure. Delta does not extract the commands automatically and relies instead on manual transcription. It is also explicitly designed for a small corpus (30 tutorials) and does not allow faceted browsing. In contrast, Sifter provides automatic command extraction and includes a browsing interface that scales to a much larger corpus (2500 tutorials). Sifter's faceted filtering and frequency-based sorting allow users to better understand how command subsequences are used across a wide variety of image manipulation tasks.

FACETED BROWSING WITH SIFTER

Sifter has three components (Figure 3). The *command extractor* takes a set of text-based tutorials as input and outputs the command-level structure of each one into a *database of command sequences*. Sifter's *browser interface* provides three different views of the collection that allow users to explore the tutorials.

The *(1) Faceted Browser View* allows users to organize, sort and filter the collection (Figure 2a). The Category facet filters the collection by a high-level tutorial category (e.g. "Photo Effects", "Web Layouts", etc.). The N-Gram facet filters the collection by the length of command subsequences. The Command Name facet allows users to filter the collection to only include tutorials that contain specific commands as chosen from a list. The order in which the user selects facets determines the hierarchical organization of the tutorials that appear in the faceted browser view. The default sort order for leaf-level tutorials is alphabetical by title. Users can optionally sort tutorials by total number of commands or total number of unique commands. The default sort order for selected N-Grams is occurrence frequency so that the most common command subsequences appear first. If tutorial Category facets have been selected, users can optionally sort command subsequences by uniqueness to the selected categories.

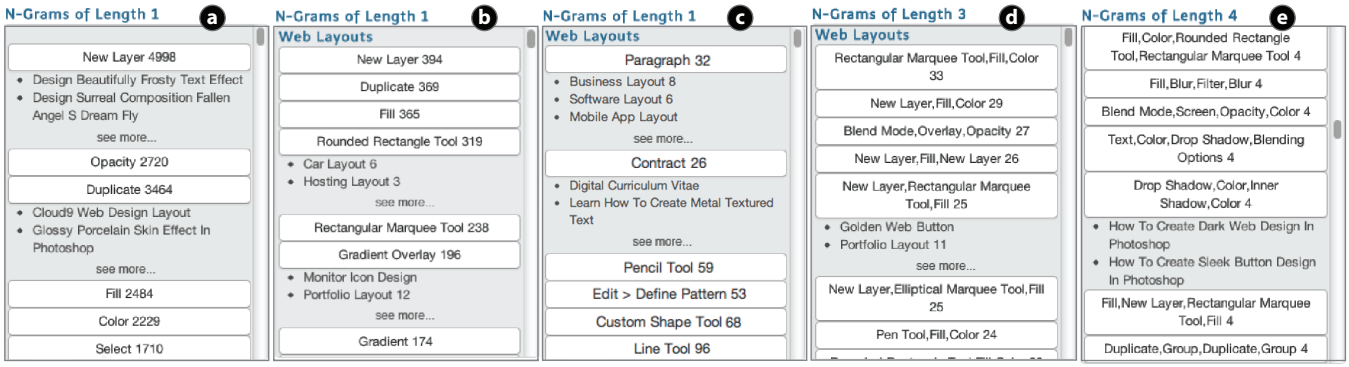

Figure 5. A user explores frequently used subsequences to learn about web layouts. He first explores 1-grams across the entire collection of tutorials (a), then limits the 1-grams to the Web Layouts category (b), then looks at 1-grams that are unique to the Web Layouts Category (c). He next considers common 3-grams (d) and 4-grams (e) for Web Layouts.

The *(2) Tutorial View* shows the original webpage for any selected tutorial as well as a table of commands that lists the commands in the tutorial (Figure 2b). The table and webpage are linked so that clicking on a command in the table scrolls the tutorial webpage to the corresponding location, while clicking a command in the webpage highlights it in the table. Users can also select a subsequence of commands in the table and perform a selection search to retrieve all tutorials containing the chosen subsequence. Finally, users can select a set of tutorials in the faceted browser and generate an *(3) Alignment View* in which each column represents the sequence of commands for a single tutorial and lines connect matching commands (Figure 2c). Sibling commands in the Photoshop menu hierarchy are given the same color (e.g. Colorize and Hue/Saturation are both colored white because they both appear within the set of Image>Adjustment commands). This view directly reveals the command-level similarities and differences between the selected set of tutorials.

We describe how a person would use the Sifter interface in the context of a user scenario.

Locating Commands within a Tutorial

Joe needs to learn how to use Photoshop to create webpage mockups and perform basic image editing for his job, but has little experience with the program. He opens Sifter and based on his interest in web design he first selects the "Web Layouts" facet. As he browses the resulting list of titles, Joe notices the "Web Logo Design" tutorial. He clicks to load it into the Tutorial View. Examining the *table of commands*, Joe sees that the tutorial makes heavy use of the Pen Tool command (Figure 4). Clicking on an instance of this command in the table scrolls the tutorial webpage to show how the command is used in context. The tutorial describes how to use the Pen Tool to draw paths. Joe can also click on a command in the tutorial webpage to highlight it in the table. Thus, the table of commands summarizes the actions required to complete the tutorial and can serve as a quick reference as he works through the tutorial. The table also allows Joe to skim through parts of the tutorial and quickly access more information about unfamiliar commands as necessary.

Browsing Frequently Used Subsequences

After looking at a few specific tutorials, Joe decides that he should first identify the commands that are most frequently used across the entire collection of tutorials. He infers that these must be the most important and useful Photoshop commands. He de-selects the "Web Layouts" category facet and sets the N-Gram facet to select single commands (N-Gram length=1). The resulting list of commands is ordered by frequency and he finds that "New Layer", "Opacity" and "Duplicate Layer" are the three most common commands (Figure 5a). He examines a few tutorials containing these commands and finds that they all involve the Photoshop layers palette and spends some time learning how layers work.

Next, Joe decides to focus on the most frequent commands within the Web Layouts category (Figure 5b). He re-selects the "Web Layouts" facet and again sets N-Gram length to 1. This time he finds that the commands "Rounded Rectangle Tool" and "Gradient Overlay" appear frequently in Web Layouts, but were not as highly ranked for the entire tutorial collection. He looks at tutorials that contain these commands and quickly learns that the "Rounded Rectangle Tool" is often used to draw buttons, while "Gradient Overlay" is commonly used to give widgets the appearance of 3D depth. To see a list of commands that are frequently used in "Web Layouts", but infrequently in the rest of the corpus, Joe uses the "Most unique to category" sort option. He finds that the "Paragraph" command is commonly used in web layout tutorials to format text while the "Contract" command is often used to shrink selections and create borders on text boxes (Figure 5c). These commands are far less common in the other Photoshop tutorial categories.

Joe then expands his search to identify repeated patterns of longer command sequences, by selecting N-Grams of lengths 3 and 4. He notices that many of the frequent three-command sequences involve a selection command followed by a fill command such as "Rectangular Marquee Tool, Fill, Color", "New Layer, Rectangular Marquee Tool, Fill" or "New Layer, Elliptical Marquee Tool, Fill", (Figure 5d). He examines some of the tutorials for each of these three-command sequences and realizes that they all describe ways to create buttons of various shapes. In the list of four-command sequences he sees several that contain styling commands such as "Drop

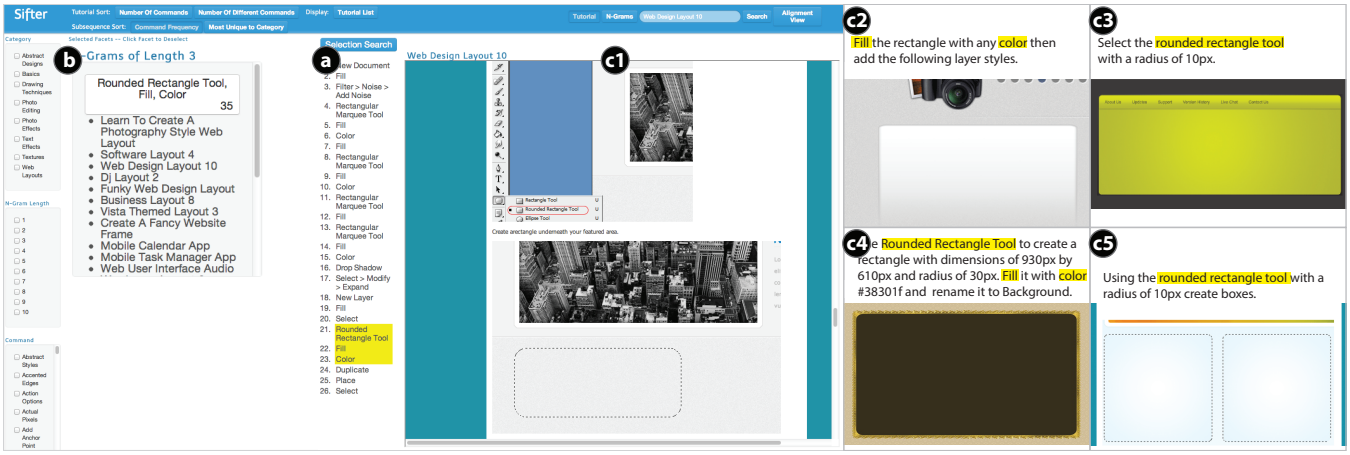

Figure 6. A user selects the command sequence used to create a content box, and clicks on the Selection Search button in the Tutorial View (a). Sifter returns a list of tutorials that contain this sequence of commands (b). The user can now click through this list of tutorials to find additional styles for content boxes (c1-c5).

Shadow", "Blending Mode", and "Blur" (Figure 5e). He learns that these command sequences all apply specific styles to the buttons. For example the 4-gram "Drop Shadow, Color, Inner Shadow, Color" adds a drop shadow and inner shadow to a button to give it a 3D appearance.

Finding Tutorials with Specific Commands

Joe decides to follow the tutorial, "How to Design a Web Layout in Photoshop", step by step. He sees that the tutorial uses "Rounded Rectangle Tool, Fill, Color" to create a box for holding pictures and text within the page layout. Joe wants to use such content boxes on his website, but does not like the style of the boxes provided in the tutorial. Joe selects the command sequence "Rounded Rectangle Tool, Fill, Color" in the table of commands (Figure 6a) and clicks the Selection Search button. Sifter returns other tutorials that also contain the selected command sequence (Figure 6b). Joe clicks through them to find additional styles for content boxes (Figure 6c). Once he finds a style he likes, he creates the content box using the new tutorial before returning to his original workflow.

Comparing Tutorials with the Alignment View

Joe finishes designing the webpage mockup and decides to try a photo manipulation tutorial. Although his hair is blonde, Joe wonders how he would look with black hair. Within the "Photo Effects" category he finds seven different hair-recoloring tutorials. He cannot decide which one to follow, so he compares them by clicking the Alignment View button.

Joe selects the tutorial titled "Hair Recoloring in Photoshop" and then flips between two layouts in the Alignment View. The *one-to-all layout* orders the tutorials (left-to-right) by their similarity to the "Hair Recoloring in Photoshop" tutorial, while the *pairwise layout* builds the ordering by incrementally choosing the next tutorial as the one that is most similar to the previous tutorial (Figure 7). In the pairwise layout, Joe immediately sees that there are three different ways to re-color hair. Tutorials a and b primarily use the "Brush Tool" and "Hue/Saturation" to complete the task. Tutorials e, f and d all use a similar four-command subsequence of ' Invert, New Layer, Fill, Blending Mode, Soft Light". Tutorial c is a combination of these two approaches, while tutorial g is completely different from the others. Since the four-command "Invert, New Layer, ..." is the most common subsequence for this task, Joe chooses the "How to Change Hair Color with Photoshop" tutorial from this group. If he had more time, Joe could try a tutorial from each of the three approaches to find the best workflow for his image.

ALGORITHMS

Sifter's faceted browsing interface relies on a set of four algorithmic techniques for extracting and comparing the command-level structure of tutorials.

Algorithm 1: Extracting Commands

To extract commands from natural language tutorials we build on the classifier of Fourney et al. [11]. Their approach is designed to identify direct references to commands (i.e., exact string matches to the command name). We extend their approach to also handle colloquial references to commands (e.g., the tutorial says to "make a mask" rather than writing the command name "Create Layer Mask", Figure 1). In a survey of 60 randomly selected tutorials we manually identified about 18% of the command references as colloquial rather than direct. Thus, accounting for such colloquial references is essential for correctly extracting commands from text-based tutorials.

Fourney et al.'s algorithm first compares each word in a tutorial to a list of command names in the Photoshop menu hierarchy using exact string matching. Photoshop directly provides this list via Edit>Keyboard Shortcuts>Summarize. We then manually extend the list to include the colloquial references we identified in the 60 tutorial survey. We also apply word stemming to automatically derive indirect references to commands. For example, applying stemming to the command name "Scale" produces the derived words "Scaling", "Scaled", etc. The final list contains direct, colloquial and indirect references to commands.

After the first stage of exact string matching, we follow the approach of Fourney et al. and apply a Naive Bayes classifier

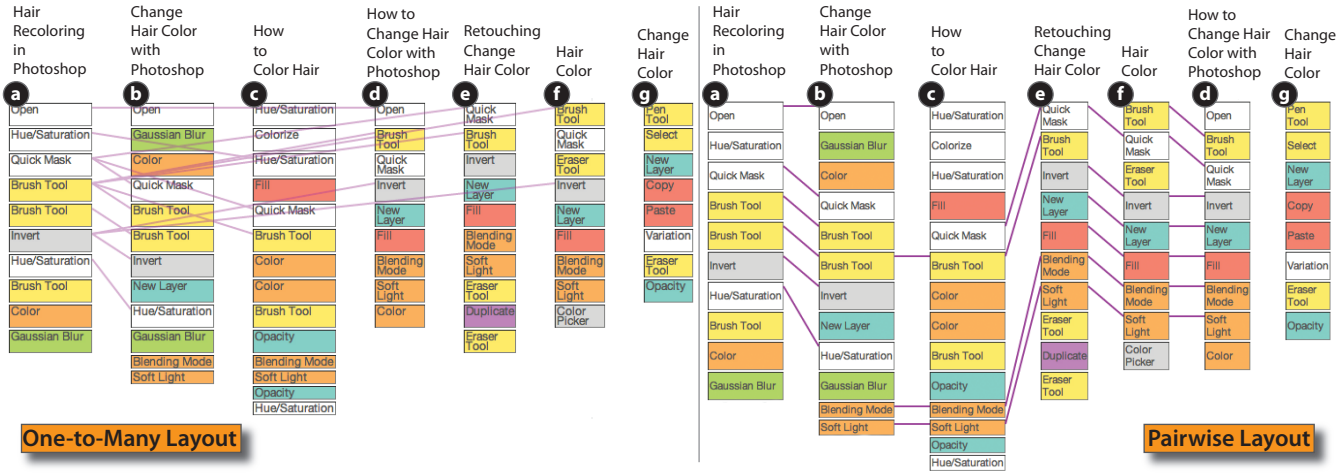

Figure 7. A user compares seven hair recoloring tutorials using the Alignment View, which displays correspondence lines between the matching commands. The *one-to-all layout* (left) orders the tutorials (left-to-right) by their similarity to the first tutorial. The *pairwise layout* (right) builds the ordering by incrementally choosing the next tutorial as the one that is most similar to the previous tutorial.

with Witten-Bell smoothing to further eliminate false positive matches. As training data we again use the 60 tutorials for which we manually marked all true command references (direct, colloquial and indirect). Using leave-one-out cross validation we obtain an average precision of 90% and recall of 99%. Applying Fourney et al.'s approach without the additional entries for colloquial and indirect references, causes the recall to drop by about 14%.

Algorithm 2: Comparing Command Sequences

We use the Needleman-Wunsch edit distance algorithm [22] to compare two sequences of tutorial commands. Like most edit-distance techniques this algorithm requires a scoring matrix that encodes the penalties for inserting, deleting and substituting commands. We use a constant penalty for insertion and deletion, and build a specialized penalty matrix for substitutions based on two objectives; (1) we increase the penalty for substituting commands that differ significantly in functionality and (2) we increase the penalty for substituting commands that are most important to the tutorial.

To identify commands with similar functionality we adopt the approach of Kong et al. [18] who observed that the Photoshop menu hierarchy groups together similar commands (e.g., all filters appear in the same part of the hierarchy). Therefore, we set the functional similarity penalty P_{FS} to the distance between commands in the menu hierarchy. To compute the importance of a command we use term-frequency-inverse document frequency (tf⋆idf) which is commonly used in information retrieval to determine the most descriptive words within a document (i.e., words that appear multiple times in the document but rarely appear in other documents). We set the importance penalty P_I to its tf⋆idf value where we treat each tutorial as a document. Then given any two commands we compute the total penalty as $P_{FS} \cdot P_I$.

Algorithm 3: Computing Frequency of Subsequences

Selecting N-Gram lengths in the faceted browser produces a list command subsequences of the corresponding lengths. By default this list of subsequences is sorted by the frequency of occurrence. To compute this frequency, we first generate every command subsequence N-Gram of length 1 to 15 for each tutorial in the collection. We then build a frequency table by hashing each resulting N-Gram and incrementing the count each time we visit the same bin. However, some Photoshop commands can be executed in arbitrary order and produce the same results. We use distance-based matching to account for such swaps. For each subsequence of length N we compute the edit distance to all other subsequences of length $N-2$ to length $N+2$ (we limit the computation to this range for efficiency). If the distance is less than a small threshold we increment the frequencies of both subsequences. We display this frequency next to the subsequence as shown in Figure 5.

Users can optionally sort the list of commands subsequences by uniqueness to a selected tutorial Category facet. For a given command subsequence we compute its category uniqueness as the number of times it occurs within the selected category, normalized by the total number of times it occurs across the collection.

Algorithm 4: Aligning Command Sequences

Our Alignment View allows users to compare a set of tutorials, so that they can easily identify the similarities and differences in their command sequences. To align the tutorials, we first establish an ordering between them. The user may select the first tutorial in this ordering which we call the *base tutorial*. If the user does not select a base then we automatically set the most representative tutorial as the base, which we compute as the centroid of the tutorial set based on edit distance.

To construct the *one-to-all layout* we first sort all the tutorials using their edit distances to the base tutorial and then display them left to right in increasing order. For the *pairwise layout* we start with the base tutorial and then iteratively choose the next tutorial as the one that is closest to the previous tutorial. The Needleman-Wunsch edit distance algorithm [22] also provides correspondences between commands (either exact matches or substitutions). Our visualization con-

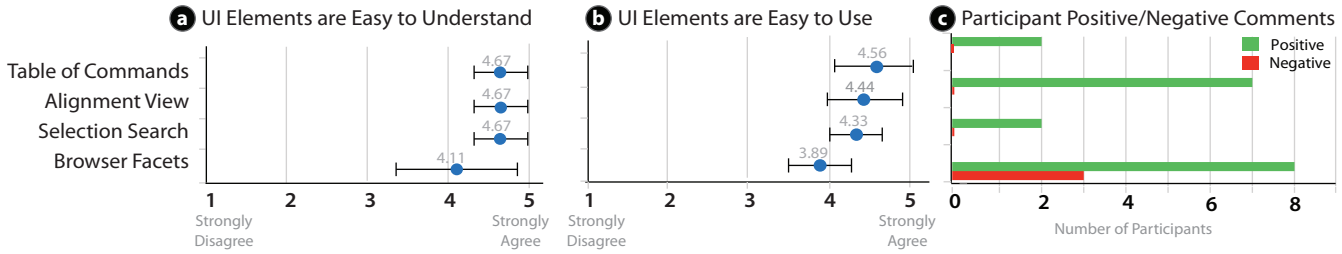

Figure 8. Participants were asked to rate if they understood each interface element (a) and if each element was easy to use (b) on a scale from 1 - Strongly Disagree to 5 - Strongly Agree. (c) Number of participants leaving positive or negative comments in the open responses.

nects matching commands with lines and uses color to represent siblings in the Photoshop menu hierarchy.

Implementation. We collected our corpus of tutorials by scraping 2500 Photoshop tutorials from eight websites over three hours. Our command extraction algorithm requires about 30 seconds per tutorial. Identifying common subsequences based on edit distance required about 7 hours for our collection. These timings are based on our unoptimized Python code, which we ran on a 2.7GHz Mac Mini with 8GB of memory. We believe that parallelizing our code to run on a cluster would significantly reduce these timings.

USER EVALUATION

We conducted an informal evaluation of Sifter with three goals: (1) to gain feedback on the usability and utility of its features, (2) to gauge user interest in exploring tutorials through command-level structure and (3) to compare task performance and user preference between Sifter and keyword search – today's status quo technique for browsing online tutorial collections.

Method

We recruited nine participants (age range 20-35), who self-rated their Photoshop expertise on a 5-point scale from novice to expert. Three rated themselves as Photoshop novices, three as intermediates and three as in-between. All participants were Computer Science students. We first led a 7-minute walk-through of the interface to briefly explain each feature of Sifter to the participants. Then, we asked the participants to complete a series of tasks using Sifter. We told them that they could abandon any task if they estimated it would take over 10 minutes to complete. We designed the tasks to exercise the four main interface elements of Sifter: browser facets, the table of commands, selection search and the alignment view. We expected the tasks would also be difficult to complete with current search interfaces. The complete set of tasks is listed in Table 1.

We asked participants to answer 5-point Likert-scale questions on their understanding of the Sifter interface elements and the ease of use of these elements. We also asked about the usefulness of a variety of browsing tasks enabled by Sifter. Finally, participants could provide longer open-ended feedback and the first author manually categorized each response about an interface element as positive or negative.

Evaluation Tasks

1. Find a common command in the category of Web Layouts.
   (Find a common command)
2. Find a command used more often in Photo Editing than in Web Layouts.
   (Compare single commands between categories)
3. Find a command unique to the category of Photo Editing.
   (Find a unique command)
4. Find two uses for the Gradient Overlay command.
   (Find different uses of a command)
5. Given one tutorial with the subsequence *Round Rectangle, Fill, Color* for creating a content box, find two more examples of this task.
   (Find multiple examples of a 3-Gram command sequence)
6. Given a task, list tutorials that follow the most common method for this task."Method" refers to a series of commands shared by one or more tutorials as shown in the Alignment View.
   (Find the most common method to perform a task)
7. Given a task, list tutorials with the three most different command flows.
   (Find different methods for a task)

Table 1. The evaluation tasks asked users to perform a variety of browsing and analysis tasks based on the command-level structure of tutorials.

User Feedback

Sifter's Interface is Easy to Understand and Use

Participants understood Sifter's interface elements and found them easy to use, giving relatively high ratings to all four of them (Figure 8a-b). However, the browser facets ranked lower than the other three elements in both understanding and ease of use. Nevertheless, eight of nine participants mentioned positive aspects of the faceted browser in the open-ended feedback (Figure 8c). The three participants who left negative comments in the open feedback mentioned low-level usability concerns; they thought the term "N-Gram" was too technical and found that filtering by facets was somewhat slow. These results suggest that participants were unfamiliar with faceted browsing based on command subsequences and found using facets to be more complex than the other interface elements. However, they recognized that facets also enable more powerful search and exploration of the tutorial collection.

Tasks Enabled by Sifter are Useful

Participants generally rated the browsing tasks enabled by Sifter to be useful (Figure 9).

In the open responses many participants mentioned that browsing and comparing tutorials based on command structure gave them a new way to approach Photoshop. Seven users mentioned they would like to use the alignment view

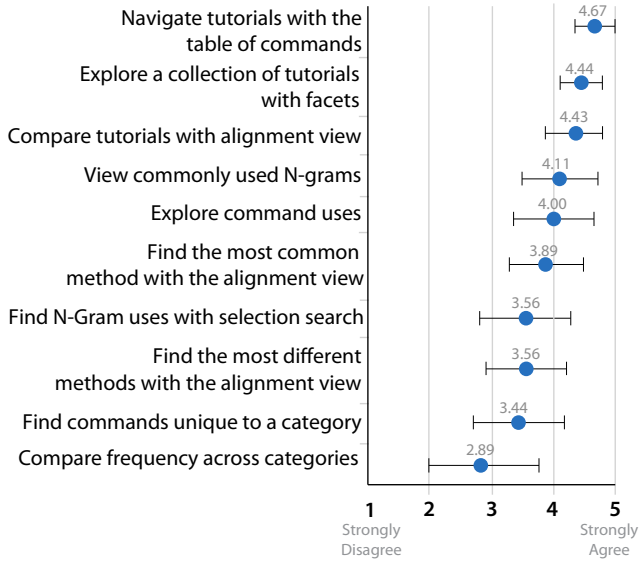

Figure 9. Participants were asked to rate usefulness of different browsing tasks on a scale from 1 - Strongly Disagree to 5 - Strongly Agree.

for assessing the similarities and differences between tutorials, finding the most common command flow to complete a task, and previewing the command flows to gauge familiarity. Five users wanted to use the Browser Facets to find more examples of unfamiliar commands and to refine tutorial search by command or domain. As one user suggested, "Sometimes I just want to know how to use one tool and I can't find a good tutorial just by Googling it". A single user was not interested in command structure as they only wanted to search by task. Nevertheless eight of the nine participants explicitly stated that they would use Sifter if they could.

Comparison of Sifter and Keyword Search

Our evaluation of Sifter was designed to assess performance on tasks we believe to be difficult with current tutorial browsing tools. To verify this intuition, we also asked our participants to execute each task (Table 1) with a keyword search interface limited to the tutorials in the Sifter corpus. By comparing task accuracy between interfaces, we evaluated whether Sifter provides additional functionality over keyword search. To create a keyword search interface we used the Google Custom Search tool [12]. We ran a within subject evaluation so that each participant performed the same set of tasks using both the Sifter and keyword search interfaces. However we changed either the concrete tutorial category or the command name mentioned in the task so that subjects couldn't directly reuse their work. To measure task accuracy we first enumerated all possible correct answers for each task. If the participant gave an answer in this set, we marked it as correct. We again told users that they could abandon tasks that would take longer than 10 minutes. Finally, we asked users to rate interface preference for each task.

Participants are More Successful and Accurate with Sifter

Participants completed all tasks with Sifter, but only 70% of the tasks with keyword search (Figure 10). They were also far more accurate with Sifter (97% vs 35% using keyword search). A one-way ANOVA finds that this difference is significant ($F(1,12) = 15.12, p = 0.0029$).

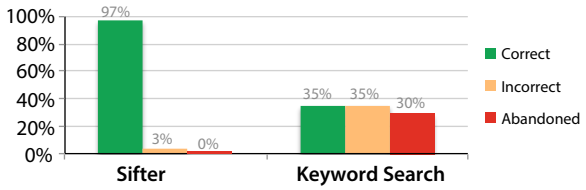

Figure 10. Correct, incorrect, and abandoned task percentages for Sifter and keyword search tasks.

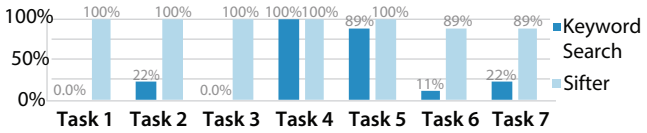

Figure 11. Percentages of correct answers for Sifter and keyword search tasks.

In the keyword search condition, participants only performed well on two of the tasks: *Find different uses of a command (Task 4)* and *Find multiple examples of a 3-Gram command sequence (Task 5)* with 100% and 89% accuracy, respectively (Figure 11). For these tasks participants directly entered either a single command or a command subsequence as the query and keyword search usually returned a set of matching tutorials. Participants then opened each tutorial and used text search within the page to find the command locations in the tutorial. All but one user preferred to use Sifter over keyword search for the evaluation tasks, even for tasks where keyword search performed well.

In aggregate, our results suggest that Sifter's interactions are accessible, useful, and provide the preferred interface for the evaluation tasks. However, we believe a larger-scale longitudinal evaluation would be necessary to rigorously verify these findings for daily work.

CORPUS ANALYSIS AND UI DESIGN IMPLICATIONS

In addition to assessing the usability and utility of Sifter, we have also analyzed the higher-level patterns in the command structure across our corpus of 2500 Photoshop tutorials. Based on this analysis we suggest several ways to extend Photoshop with a more efficient or task specific user interface.

Using Sifter we have found that each tutorial category contains some commands that appear more frequently within that category (Table 2). For example, 79% of the uses of the "Clone Stamp Tool" appear in tutorials within the "Photo Manipulation" category, but only 9% appear within the "Web Layouts" category. This tool enables photo retouching and is therefore less useful in the context of web design. Similarly, some command subsequences occur primarily within particular categories. For example, the three-command subsequence "Adjustment Layers, Clipping Mask, Black & White" occurs only within the "Photo Manipulation" category and does not appear in any other category. This subsequence selectively desaturates the image and thereby highlights the remaining colored region.

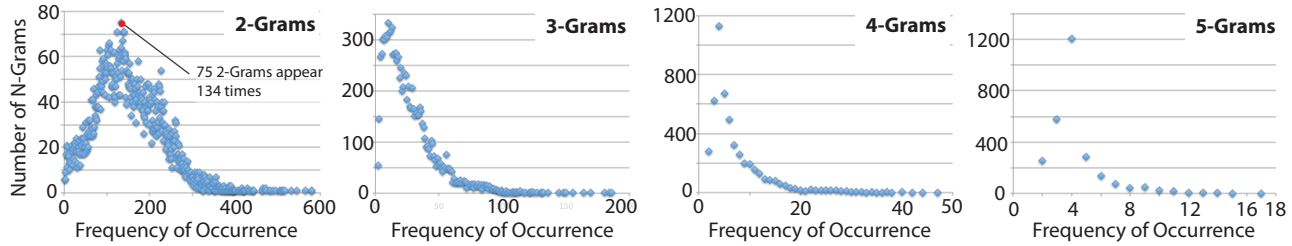

Figure 12. Number of N-Grams that appear at each frequency greater than one for N equals 2 to 5.

Command(s)	Drawing/ Painting	Web Layouts	Photo Manipulation	Text Effects
Clone Stamp Tool	9	9	79	3
Perspective	22	14	61	4
Adjustment Layers, Clipping Mask, Black & White	0	0	100	0
Copy, Paste, New Layer	15	18	55	12
Select, Rounded Rectangle Tool, Fill	0	100	0	0
New Layer, Pen Tool, Fill	10	40	50	0

Table 2. Examples command sequences and their distribution across the different tutorial categories.

This data suggests that for users who primarily work on problems that fall into one category it may be worth designing custom UI panels that contain the most commonly used command subsequences. Quick access to these common command strategies could simplify workflows and help novice users learn the most efficient way to complete tasks.

We have also analyzed the frequency of all N-Grams of length 2 to 5 across the complete tutorial corpus. In Figure 12 we plot the number of N-Grams that appear at each frequency greater than one. For example, 75 different 2-grams appear 134 times. Not surprisingly, the rightmost tails of these plots are low, indicating that there are few command subsequences that appear at high frequency. Since these are the most common N-Grams and there are relatively few of them, novice users could focus on learning these command subsequences first to quickly become proficient with Photoshop. We also find that the number of N-Grams that appear at the lowest frequencies is relatively small, indicating that if an N-Gram appears in a tutorial, it is likely to appear many times in other tutorials. Since many command patterns occur frequently, it may be possible to watch users as they execute commands and infer the command-level strategies they are applying. The Photoshop interface could then highlight the next commands in the sequence, suggest related tutorials for the task, or notify users if they deviate from a sequence.

CONCLUSION

We have presented Sifter, an interface that allows users to browse and analyze a large collection of image manipulation tutorials based on their command-level structure. To build this interface, we first extract the command sequences from a collection of 2500 tutorials. We then provide an interface with three views; a (1) Faceted Browser View for organizing, filtering and sorting the tutorials by their commands, a (2) Tutorial View for examining individual tutorials and an (3) Alignment View for comparing the similarities and differences in the command structure between a subset of tutorials. User feedback suggests that Sifter's interface is relatively easy to understand and use. Users also recognized that the faceted browser enables a new way to search and explore a tutorial collection based on command-level structure. As more and more instructional material appears online, we believe that making use of the underlying command-level structure will be essential for helping people learn software tools.

REFERENCES

1. AutoDesk, Inc. Project Chronicle. `https://chronicle.autodesk.com/`, 2013.
2. L. Bergman, V. Castelli, T. Lau, and D. Oblinger. Docwizards: a system for authoring follow-me documentation wizards. In *Proceedings of UIST*, pages 191–200. ACM, 2005.
3. 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 Transactions on Graphics (TOG)*, 30(5):120:1–120:14, Oct. 2011.
4. J. M. Carroll. *The Nurnberg funnel: designing minimalist instruction for practical computer skill*. MIT Press, Cambridge, MA, USA, 1990.
5. H. Chen, L. Wei, and C. Chang. Nonlinear revision control for images. *ACM Transactions on Graphics (TOG)*, 30(4):105, 2011.
6. P. Chi, S. Ahn, A. Ren, M. Dontcheva, W. Li, and B. Hartmann. Mixt: Automatic generation of step-by-step mixed media tutorials. In *Proceedings of UIST*, pages 93–102. ACM, 2012.
7. P. Chilana, A. J. Ko, and J. O. Wobbrock. Lemonaid: Selection-based crowdsourced contextual help for web applications. In *Proceedings of CHI*, pages 1549–1558, 2012.
8. J. Denning, W. Kerr, and F. Pellacini. Meshflow: interactive visualization of mesh construction sequences. *ACM Transactions on Graphics (TOG)*, 30(4):66, 2011.
9. M. Dixon and J. Fogarty. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of CHI*, pages 1525–1534. ACM, 2010.

10. M. Ekstrand, W. Li, T. Grossman, J. Matejka, and G. Fitzmaurice. Searching for software learning resources using application context. In *Proceedings of UIST*, pages 195–204. ACM, 2011.

11. A. Fourney, B. Lafreniere, R. Mann, and M. Terry. "Then click 'OK!'" extracting references to interface elements in online documentation. In *Proceedings of CHI*, pages 35–38, 2012.

12. Google, Inc. Google Custom Search Engine. `https://www.google.com/cse`, 2013.

13. F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi. Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics*, 28(3):66:1–66:9, July 2009.

14. T. Grossman, G. Fitzmaurice, and R. Attar. A survey of software learnability. In *Proceedings of CHI*, pages 649–658. ACM Press, 2009.

15. T. Grossman, J. Matejka, and G. Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of UIST*, pages 143–152. ACM, 2010.

16. J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1189–1196, 2008.

17. C. Kelleher, C. Forlines, and R. Pausch. Stencil-based help and tutorials. In *Proceedings of CHI*, pages 541–550, 2005.

18. N. Kong, T. Grossman, B. Hartmann, G. Fitzmaurice, and M. Agrawala. Delta: A tool for representing and comparing workflows. In *Proceedings of CHI*, pages 1027–1036, 2012.

19. B. Lafreniere, A. Bunt, M. Lount, F. Krynicki, and M. A. Terry. AdaptableGIMP: designing a socially-adaptable interface. In *Adjunct Proceedings of UIST*, pages 89–90. ACM, 2011.

20. G. Laput, E. Adar, M. Dontcheva, and W. Li. Tutorial-based interfaces for cloud-enabled applications. In *Proceedings of UIST*, pages 113–122. ACM, 2012.

21. T. Lau, C. Drews, and J. Nichols. Interpreting written how-to instructions. In *Proceedings of IJCAI*, pages 1433–1438. Morgan Kaufmann Publishers Inc., 2009.

22. C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

23. J. Matejka, T. Grossman, and G. Fitzmaurice. Ip-qat: in-product questions, answers, & tips. In *Proceedings of UIST*, pages 175–184. ACM, 2011.

24. J. Matejka, W. Li, T. Grossman, and G. Fitzmaurice. Communitycommands: command recommendations for software applications. In *Proceedings of UIST*, pages 193–202. ACM, 2009.

25. T. Nakamura and T. Igarashi. An application-independent system for visualizing user operation history. In *Proceedings of UIST*, pages 23–32. ACM, 2008.

26. S. Pongnumkul, M. Dontcheva, W. Li, J. Wang, L. Bourdev, S. Avidan, and M. F. Cohen. Pause-and-play: automatically linking screencast video tutorials with applications. In *Proceedings of UIST*, pages 135–144. ACM, 2011.

27. V. Ramesh, C. Hsu, M. Agrawala, and B. Hartmann. Showmehow: translating user interface instructions between applications. In *Proceedings of UIST*, pages 127–134. ACM, 2011.

28. M. Rettig. Nobody reads documentation. *Communications of the ACM*, 34(7):19–24, July 1991.

29. S. Su. *Enhanced Visual Authoring Using Operation History*. PhD thesis, Massachusetts Institute of Technology, 2009.

30. T. Yeh, T. Chang, and R. Miller. Sikuli: using gui screenshots for search and automation. In *Proceedings of UIST*, pages 183–192. ACM, 2009.